Available online at www.sciencedirect.com
ScienceDirect
Discrete Optimization 4 (2007) 77–86
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

Improving the feasibility pump

Tobias Achterberg*, Timo Berthold

Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany

Received 23 September 2005; received in revised form 14 April 2006; accepted 6 October 2006
Available online 18 December 2006

Abstract

The feasibility pump described by Fischetti, Glover, and Lodi [M. Fischetti, F. Glover, A. Lodi, The feasibility pump, Mathematical Programming 104 (1) (2005) 91–104] and Bertacco, Fischetti, and Lodi [L. Bertacco, M. Fischetti, A. Lodi, A feasibility pump heuristic for general mixed-integer problems, Technical Report OR/05/5, DEIS – Università di Bologna, Italy, May 2005] has proved to be a very successful heuristic for finding feasible solutions of mixed integer programs. The quality of the solutions in terms of the objective value, however, is sometimes quite poor. This paper proposes a slight modification of the algorithm in order to find better solutions. Extensive computational results show the success of this variant: for 89 out of 121 MIP instances the modified version produces improved solutions in comparison to the original feasibility pump.

Keywords: Mixed integer programming; Primal heuristics; Feasibility pump

1. Introduction

A mixed-integer program (MIP) can be stated as

$$\text{(MIP)} \qquad \min\{c^T x \mid Ax \le b, l \le x \le u, x_j \in \mathbb{Z} \text{ for all } j \in I\}$$

with $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c, l, u \in \mathbb{R}^n$, and $I \subseteq N = \{1, \ldots, n\}$. Furthermore, let $B \subseteq I$ be the set of binary variables. For solving such problems it is important to quickly find feasible solutions of high quality: a good primal bound helps to cut off suboptimal branches in the search tree of a branch-and-bound based algorithm, and improvement heuristics such as Local Branching [9], guided dives, and RINS [7] can only be applied after a feasible solution has been found.

Several heuristic methods to produce feasible solutions for MIP have been proposed in the literature, including Hillier [14], Balas and Martin [4], Saltzman and Hillier [19], Glover and Laguna [10–12], Løkketangen and Glover [16], Glover et al. [13], Nediak and Eckstein [18], and Balas et al. [3,5].

The so-called feasibility pump was proposed by Fischetti, Glover, and Lodi [8] and improved by Bertacco, Fischetti, and Lodi [6]. This heuristic turned out to be very successful in finding feasible solutions even for very hard MIP instances. However, the quality of these solutions in terms of their objective value is sometimes poor, which was observed in [6] for MIPs with general integer variables and which we also confirm in the computational results of Section 3.

* Corresponding address: Zuse Institute Berlin, Optimization, Takustr. 7, 14195 Berlin, Germany.
E-mail addresses: achterberg@zib.de (T. Achterberg), berthold@zib.de (T. Berthold).

1572-5286/$ - see front matter
doi:10.1016/j.disopt.2006.10.004

This paper suggests a slight modification of the feasibility pump. In contrast to the original version, the modified *objective feasibility pump* takes the objective function c of the MIP into account during the course of the algorithm. Computational results show that the solution quality can indeed be improved by our approach without losing the ability to find feasible solutions in a reasonable amount of time.

The rest of the paper is organized as follows. The remainder of Section 1 reviews the original version of the feasibility pump as described by Bertacco, Fischetti, and Lodi [6]. Section 2 introduces the modifications included in the objective feasibility pump. Finally, Section 3 gives computational results on a large test set of 121 MIP instances from the MIPLIB 2003 [2], the Mittelmann test set [17], and instances from Danna et al. [7].

1.1. The feasibility pump

The feasibility pump heuristic proceeds as follows: First the LP relaxation

$$\text{(LP)} \qquad \min\{c^T x \mid Ax \le b, l \le x \le u\}$$

of (MIP) is solved. Then, for $S = B$ or $S = I$ (see Section 1.2), the LP solution $x^\star$ is rounded to a vector $\tilde{x} = [x^\star]^S$, with $[\cdot]^S$ defined by

$$[x]^S_j := \begin{cases} \lfloor x_j + 0.5 \rfloor & \text{if } j \in S \\ x_j & \text{if } j \notin S. \end{cases} \tag{1}$$

If $\tilde{x}$ is not feasible, an additional LP is solved in order to find a new point in the LP polyhedron

$$P := \{x \in \mathbb{R}^n \mid Ax \le b, l \le x \le u\}$$

which is, w.r.t. the integer variables of S, closest to $\tilde{x}$, i.e., that minimizes

$$\Delta^S(x, \tilde{x}) := \sum_{j \in S} |x_j - \tilde{x}_j|.$$

The procedure is iterated using this point as new solution $x^\star \in P$. Thereby, the algorithm creates two sequences of points: one with points $x^\star$ that fulfil the inequalities, and one with points $\tilde{x}$ that fulfil the integrality requirements. The algorithm terminates if the two sequences converge or if a predefined iteration limit is reached.

In order to determine a point

$$x^\star := \operatorname{argmin} \{\Delta^S(x, \tilde{x}) \mid x \in P\} \tag{2}$$

in P, which is nearest to $\tilde{x}$, the following LP is solved:

$$\begin{array}{lll} \min & \displaystyle\sum_{j \in S: \tilde{x}_j = l_j} (x_j - l_j) + \sum_{j \in S: \tilde{x}_j = u_j} (u_j - x_j) + \sum_{j \in S: l_j < \tilde{x}_j < u_j} d_j & \\ \text{s.t.} & Ax \le b & \\ & d \ge x - \tilde{x} & \\ & d \ge \tilde{x} - x & \\ & l \le x \le u. & \end{array} \tag{3}$$

The variables d_j are introduced to model the nonlinear function $d_j = |x_j - \tilde{x}_j|$ for integer variables x_j that are not equal to one of their bounds in the rounded solution $\tilde{x}$.

1.2. Implementational issues

In the course of the algorithm, two main problems arise: First, the procedure can be caught in a cycle. That means, the same sequence of integer and LP-feasible points is visited over and over again. Second, the progress in driving the integer points towards feasibility might be very slow. In [8,6] the first problem is handled by performing a so-called *restart* each time an integer point $\tilde{x}$ is generated that was already visited in a prior iteration. In a restart a random perturbation step is executed, which shifts some of the variables randomly up or down and installs this perturbed vector as a new integer point $\tilde{x}$ to continue the search.

To handle the second issue, the feasibility pump algorithm as described by Bertacco, Fischetti, and Lodi [6] is subdivided into three stages. At the first stage, a couple of iterations (so-called *pumping rounds*) are performed just on the binary variables $S = B$ by relaxing the integrality conditions on the general integer variables. The first stage is stopped

- after an LP solution $x^\star$ was found with all binary variables being integral,
- the 'fractionality' measure
$$f^S(x^\star) := \sum_{j \in S} f(x_j^\star) \quad \text{with } f(x_j^\star) := |x_j^\star - \lfloor x_j^\star + 0.5 \rfloor| \quad \text{and} \quad S = B$$
could not be decreased by at least $p = 10\%$ in a certain number of iterations, or
- a pumping round limit is reached.

If the first stage does not yield a feasible solution, the second stage invokes pumping rounds taking all integer variables $S = I$ into account. As initial integer point $\tilde{x}$ one chooses a point visited in Stage 1 which was closest to the LP polyhedron. The second stage is aborted for analogous reasons as Stage 1 (using different parameter settings), or if 100 restarts have been performed in Stage 2.

If still no solution is found, a third stage is executed. Using a point $\tilde{x}$ from Stage 2 closest to P, the MIP

$$\min\{\Delta^I(x, \tilde{x}) \mid Ax \leq b, l \leq x \leq u, x_j \in \mathbb{Z} \text{ for all } j \in I\} \tag{4}$$

is processed by a MIP solver which stops after the first feasible solution is found. One expects that the nearly feasible point $\tilde{x}$ has integer feasible points in its vicinity. It is therefore likely that the MIP solver finds a feasible solution early in the search process if the objective function of (4) is used.

2. The objective feasibility pump

Finding a high-quality solution of a MIP means to find a point $x \in \mathbb{R}^n$ satisfying three conditions: $x \in P$, $x_j \in \mathbb{Z}$ for all $j \in I$, and $c^T x$ is as small as possible. The feasibility pump generates sequences of points fulfilling the first and the second criteria, respectively, hopefully resulting in a point that satisfies both. However, despite the computation of the starting point, which is chosen to be the optimum of the LP relaxation, the third condition is disregarded. Therefore, the solution quality is usually rather poor; see Section 3.

For 0-1 MIPs Fischetti, Glover, and Lodi [8] address this issue by updating an objective limit each time a new solution has been found and calling the feasibility pump again on this restricted MIP. For MIPs with general integer variables Bertacco, Fischetti, and Lodi [6] propose to apply local search strategies such as Local Branching [9] or RINS [7] to improve the solution.

We take a different approach. Instead of instantly discarding the original objective function of the MIP, we gradually reduce its influence and increase the weight of the artificial objective function $\Delta^S(\cdot, \tilde{x})$ of the feasibility pump. The hope is that we still converge to a feasible solution but concentrate the search on the region of high-quality points.

In the remainder of the paper we assume $c \neq 0$. Our modification of the feasibility pump replaces the distance function $\Delta^S(\cdot, \tilde{x})$ by a convex combination of $\Delta^S(\cdot, \tilde{x})$ and the original objective function vector c:

$$\Delta_\alpha^S(x, \tilde{x}) := (1 - \alpha)\Delta^S(x, \tilde{x}) + \alpha \frac{\sqrt{|S|}}{\|c\|} c^T x \quad \text{with } \alpha \in [0, 1]. \tag{5}$$

Here, $\|\cdot\|$ is the Euclidean norm of a vector. Note that $\sqrt{|S|}$ is the Euclidean norm of the objective function vector of (3). We compute $x^\star$ using Δ_α^S instead of Δ^S in (2) by appropriately modifying the objective function of (3). In each pumping round α is geometrically decreased with a fixed factor $\varphi \in (0, 1)$, i.e., $\alpha_{t+1} = \varphi\, \alpha_t$ and $\alpha_0 \in [0, 1]$. With increasing iteration index t, this puts the emphasis more and more towards feasibility and decreases the influence of the original objective function. Note that you can obtain the original feasibility pump by choosing $\alpha_0 = 0$.

The introduction of Δ_α^S requires a modification of the cycle detection step in the feasibility pump algorithm. Especially during the first pumping rounds it might happen frequently that integer points are revisited, because of the higher resemblance of subsequent functions $\Delta_\alpha^S(\cdot, \tilde{x})$. Reaching the same point once again, however, does not automatically imply that the process runs into an infinite cycle as in the original heuristic. This is because we are now using different directions $\Delta_{\alpha_t}^S$ in different iterations t in contrast to the old Δ^S, which only depends on the

current integer point $\tilde{x}$. After α was decreased sufficiently, it is likely that the algorithm leaves the cycle. We therefore remember the visited points as pairs $(\tilde{x}, \alpha_t)$ and conduct a restart at iteration t only if the point $\tilde{x}$ was already visited at iteration $t' < t$ with $\alpha_{t'} - \alpha_t \leq \delta_\alpha$ and $\delta_\alpha \in [0, 1]$ being a fixed parameter value.

Instead of modifying the cycling detection as described above, one could also use different concepts such as tabu lists (see, e.g., Glover and Laguna [12]). However, we tried to retain the proceeding of the original feasibility pump as close as possible in order to obtain good comparability between the two versions.

The pseudocode of the modified feasibility pump reads as follows.

Algorithm 2.1 (*Objective Feasibility Pump*).

Stage 1:

1. Initialize $x^\star := \text{argmin}\{c^T x \mid x \in P\}$, $S := B$, $\tilde{x} := [x^\star]^S$, $t := 0$, $maxIter := maxIter_{ST1}$, $maxStalls := maxStalls_{ST1}$, $restarts := 0$, $L := \emptyset$.
2. If $\tilde{x}$ did not change since the last iteration, round the T most fractional variables $x^\star_j$, $j \in S$, to the other side compared to $\tilde{x}_j$ (with T being a parameter).
3. While there exists $(\tilde{x}', \alpha_{t'}) \in L$ with $\tilde{x}' = \tilde{x}$ and $\alpha_{t'} - \alpha_t \leq \delta_\alpha$, perform a random perturbation on $\tilde{x}$ (see [6]) and set $restarts := restarts + 1$.
4. If $\tilde{x}$ is feasible for (MIP) $\rightarrow$ stop.
5. Set $L := L \cup \{(\tilde{x}, \alpha_t)\}$. Set $t := t + 1$, $\alpha_t := \varphi\alpha_{t-1}$.
6. If $t > maxIter$, go to next stage.
7. Solve $x^\star := \text{argmin}\{\Delta^S_{\alpha_t}(x, \tilde{x}) \mid x \in P\}$.
8. If $x^\star = \tilde{x}$, go to next stage.
9. If $f^S(x^\star)$ did not decrease by at least a fraction $p \in (0, 1)$ in the last $maxStalls$ pumping rounds, go to next stage.
10. Set $\tilde{x} := [x^\star]^S$. Go to Step 2.

Stage 2:

1. Initialize $\tilde{x}$ to be an integer point of Stage 1 with minimal $\Delta^B(x, \tilde{x})$, $x \in P$. Set $t := 0$, $S := I$, $maxIter := maxIter_{ST2}$, $maxStalls := maxStalls_{ST2}$, $restarts := 0$, $L := \emptyset$.
2. Perform Steps 2–10 of Stage 1, but if $restarts > maxRestarts$ in Step 3, go to Stage 3, and if $x^\star = \tilde{x}$ in Step 8, stop.

Stage 3:

1. Solve MIP (4) with $\tilde{x}$ being an integer point of Stage 2 with minimal $\Delta^I(x, \tilde{x})$, $x \in P$. Stop after the first feasible solution has been found.

3. Computational results

This section compares the performances of the feasibility pump described in [6] and the objective feasibility pump described in this paper. We thank Livio Bertacco, Matteo Fischetti, and Andrea Lodi for sending us the source code of their original version, in which we incorporated our ideas, thereby making a direct comparison possible. All computations were performed on a 3.4 GHz Pentium-4 with 512 KB cache and 3 GB RAM. CPLEX 9.03 [15] was used as the underlying LP solver. We set a time limit of one hour in all runs.

3.1. Test set and settings

The computations were performed on a wide test set consisting of 121 instances taken from

- MIPLIB 2003 [2],
- the MIP collection of Mittelmann [17], and
- the instances used in [6], which are described in [7].

In all runs we used the parameter settings for the feasibility pump as suggested in [6], as follows. The maximum number of total iterations for Stages 1 and 2 were set to $maxIter_{ST1} = 10\,000$ and $maxIter_{ST2} = 2000$. The maximum number of iterations without a pumping cycle of at least $p = 10\%$ improvement were set to $maxStalls_{ST1} = 70$ and $maxStalls_{ST2} = 600$. The shifting in Step 2 is applied on a random number of $T \in [10, 30]$ variables, but only

variables x_j with current fractionality $f(x_j^\star) > 0.02$ are regarded as shifting candidates. For the unmodified feasibility pump we set $\alpha_0 = 0$, so as to deactivate all modifications. For the objective feasibility pump we set $\alpha_0 = 1$, $\varphi = 0.9$, and $\delta_\alpha = 0.005$.

Because we wanted to investigate the performance of the feasibility pump used as a root node heuristic inside a MIP solver, we applied the MIP preprocessing of CPLEX prior to running the feasibility pump algorithm itself. This usually avoids difficulties with the scaling of degenerated objective functions in the modified distance function Δ_α^S, see Eq. (5). For example, some instances in our test set have objective functions consisting of only a single artificial variable which is defined as a linear combination of several other variables by an equality constraint. Such equations lead to unbalanced situations in the calculation of Δ_α^S, since in this case the norm $\|c\|$ of the objective function is misleading. We observed that MIP preprocessing usually resolves this issue.

We also applied CPLEX 9.03 to the problem instances in order to compare the feasibility pump to the primal heuristics of a state-of-the-art MIP solver. We deactivated all cutting planes as they are also not used in the feasibility pump, and stopped CPLEX after the root node was solved and the heuristics were applied. Note that in contrast to Fischetti, Glover, and Lodi [8] we report the *best* solution found in the root node instead of the *first*. Since we want to evaluate the quality of the solutions obtained by the feasibility pump it seems fair to compare against *all* CPLEX root node heuristics. However, this enables CPLEX also to apply improvement heuristics such as RINS [7].

3.2. Results

Table 1 compares the performance of the two feasibility pump versions. The left-hand side shows the results of the original version proposed by Bertacco, Fischetti, and Lodi [6]. The central column shows the results of the objective feasibility pump described in this paper. As an additional comparison the results of CPLEX's root node heuristics are displayed in the right-hand side column. The column 'Objective' contains the objective values of the feasible solutions that were found with either algorithm. A bar '–' means that no solution was found within the time limit of one hour, or, in the case of CPLEX, before branching took place. Values marked with a star '$\star$' indicate instances where the heuristic found an optimal solution. 'Gap' denotes the percentage gap γ to the optimal or best-known solution of the corresponding instance. It is printed in bold face if the corresponding version of the feasibility pump produced a solution with a better or equal value than the other version. The gap is calculated as

$$\gamma := 100 \cdot \frac{(\tilde{c} - c^\star)}{|c^\star|}$$

with $\tilde{c}$ being the value of the heuristic solution and $c^\star$ being the optimal or best-known solution value of the instance. If $\tilde{c} = c^\star = 0$ we define $\gamma := 0$. If $\tilde{c} > c^\star$ and $c^\star = 0$ we define $\gamma := \infty$. The instances displayed in italics in the tables are those for which we do not know the optimal solution. In this case we compare to the best solution we know, which was either generated by CPLEX 9.1 running for an hour with default settings, retrieved from the MIPLIB 2003 web site [1], or produced by one of the feasibility pump versions.

Column 'Time' shows the time in seconds to find a solution, or, in the case of CPLEX, to process the root node.

The first geometric means at the bottom of Table 1 are calculated over the 116 instances for which a solution was found by both versions of the feasibility pump. The second geometric means are calculated over the 77 instances for which all three algorithms, including CPLEX, found a solution. In the calculations of the geometric means individual values smaller than 1 are replaced by 1.

As one can see in the table the objective feasibility pump produced a strictly better solution than the original version in 89 out of 121 cases, whereas the unmodified feasibility pump ranked first in 17 cases. For 11 instances both versions computed the same objective value, and for four instances both versions were not able to find any feasible solution within one hour. Only for one instance, namely `acc-6`, the original version could find a solution while the objective feasibility pump did not succeed.

The running times usually differ only slightly in terms of absolute numbers. Only on some instances the objective feasibility pump was significantly slower, namely on `air04`, `dano3mip`, `momentum3`, `mzzv42z`, `rd-rplusc-21`, the three `dano` instances, on `qap10`, `acc-6`, and `neos16`. However, for all those instances except `acc-6` a better solution was found. The objective feasibility pump was substantially faster on `momentum1`, `protfold`, `t1717`, `icir97_potential`, and `neos10`. Nevertheless, the solutions on these instances are at least as good as the ones of the unmodified feasibility pump. The quality improvement can also be seen in the geometric means: the mean

Table 1
Comparison of original feasibility pump and objective sensitive version

Name	Original feasibility pump			Objective feasibility pump			CPLEX 9.03		
	Objective	Gap %	Time	Objective	Gap %	Time	Objective	Gap %	Time
10teams	958	4	1	952	**3**	5	–	–	0
a1c1s1	17762	53	1	16076.6	**39**	1	21029.4	81	0
aflow30a	2549	**120**	0	4105	254	0	1304	13	0
aflow40b	7682	558	0	2049	**75**	0	–	–	0
air04	58608	4	15	57298	**2**	164	56843	1	5
air05	30883	17	3	26942	**2**	8	27578	5	2
arki001	7.75064e+06	2	12	7.70474e+06	**2**	10	–	–	0
atlanta-ip	166.014	75	45	138.012	**45**	56	–	–	95
cap6000	−2.37503e+06	3	0	−2.42701e+06	**1**	0	−2.44946e+06	0	0
dano3mip	1000	43	30	769.25	**10**	383	761.9	9	145
danoint	93	42	1	87	**32**	3	66.5	1	0
disctom	−5000⋆	**0**	9	−5000⋆	**0**	11	–	–	16
ds	–	–	3600	–	–	3600	615.992	117	107
fast0507	245	41	20	179	**3**	21	177	2	39
fiber	4.01694e+06	890	0	1.20751e+06	**197**	0	471805	16	0
fixnet6	9283	133	0	4807	**21**	0	4435	11	0
gesa2-o	4.91411e+07	91	0	2.6504e+07	**3**	0	–	–	0
gesa2	2.82478e+07	10	1	2.67652e+07	**4**	1	2.58091e+07	0	0
glass4	5.20005e+09	333	0	3.10003e+09	**158**	0	–	–	0
harp2	−6.06939e+07	**18**	0	−5.58762e+07	24	0	−7.31719e+07	1	0
liu	6378	444	0	4100	**250**	1	5268	349	0
manna81	−12891	2	0	−12894	**2**	0	−13164⋆	0	3
markshare1	362	36100	0	194	**19300**	1	1095	109400	0
markshare2	1523	152200	0	365	**36400**	0	346	34500	0
mas74	18692.3	**58**	0	19033.1	61	0	14372.9	22	0
mas76	72860.6	82	0	50124	**25**	0	40005.1⋆	0	0
misc07	4100	46	1	3425	**22**	0	–	–	0
mkc	−288.01	49	0	−289.95	**49**	0	−499.464	11	0
mod011	−2.38751e+07	56	0	−4.56201e+07	**16**	1	−5.14737e+07	6	0
modglob	3.08143e+07	49	0	2.10876e+07	**2**	0	2.07868e+07	0	0
momentum1	359238	229	818	346535	**218**	223	–	–	26
momentum2	–	–	3600	–	–	3600	–	–	40
momentum3	509585	38	272	420724	**14**	599	–	–	2229
msc98-ip	3.02737e+07	30	34	3.02655e+07	**30**	38	–	–	57
mzzv11	−11286	48	118	−17688	**19**	112	–	–	59
mzzv42z	−12472	39	22	−15470	**25**	78	–	–	33
net12	337	**57**	8	337	**57**	14	–	–	37
noswot	−26	37	0	−40	**2**	0	−40	2	0
nsrand-ipx	78240	**53**	0	89120	74	1	56000	9	0
nw04	19124	13	3	17856	**6**	9	17056	1	1
opt1217	−16⋆	**0**	0	−16⋆	**0**	0	−16⋆	0	0
p2756	91972	2844	2	89266	**2757**	3	3555	14	0
pk1	78	**609**	0	83	655	0	31	182	0
pp08a	12180	66	1	10940	**49**	0	8070	10	0
pp08aCUTS	10750	46	0	8530	**16**	0	8050	10	0
protfold	−10	67	683	−12	**60**	268	–	–	11
qiu	1945.5	1564	0	625.709	**571**	0	94.6865	171	0
rd-rplusc-21	173065	1	375	171182⋆	**0**	790	–	–	15
roll3000	18812	**46**	0	24417.6	89	6	–	–	0
rout	1720.82	**60**	0	1773.95	65	0	1768.21	64	0
set1ch	72987.8	**34**	1	84167.5	54	0	67334.5	23	0
seymour	527	25	1	445	**5**	3	435	3	10
sp97ar	9.57074e+08	44	3	9.40566e+08	**42**	3	6.75288e+08	2	13
stp3d	–	–	3600	–	–	3600	–	–	2036
swath	1630.8	192	3	1280.95	**130**	13	1405.58	152	0

Table 1 (*continued*)

Name	Original feasibility pump			Objective feasibility pump			CPLEX 9.03		
	Objective	Gap %	Time	Objective	Gap %	Time	Objective	Gap %	Time
t1717	237564	23	556	195779	**1**	171	–	–	27
timtab1	1.51227e+06	98	1	1.33858e+06	**75**	1	–	–	0
timtab2	1.91798e+06	58	1	1.73262e+06	**42**	4	–	–	0
tr12-30	269910	107	0	163794	**25**	0	–	–	0
vpm2	19.5	42	0	15.25	**11**	0	16.25	18	0
bell3a	9.85707e+07	11121	0	7.21256e+07	**8111**	0	1.67409e+08	18958	0
bell5	4.81498e+07	437	0	4.08948e+07	**356**	0	9.50297e+07	960	0
gesa3	3.5368e+07	26	0	2.89813e+07	**4**	1	2.80236e+07	0	0
gesa3_o	6.76543e+07	142	1	2.87697e+07	**3**	0	2.79914e+07	0	0
l152lav	4781	1	0	4757	**1**	0	4755	1	0
stein45	45	50	0	35	**17**	0	31	3	0
ran8 × 32	6033	15	0	5817	**11**	0	5553	6	0
ran10 × 26	5050	18	0	4833	**13**	0	4745	11	0
ran12 × 21	4330	18	0	4231	**15**	0	4080	11	0
ran13 × 13	3705	**14**	0	3820	17	0	3517	8	0
binkar10_1	7170.23	6	0	7156.21	**6**	1	6874.2	2	0
eilD76	1616.97	**83**	10	2300.06	160	10	1196.97	35	0
irp	12715.3	5	2	12162.4	**0**	1	12162.4	0	0
mas284	105336	15	0	99522.7	**9**	0	93708.1	3	0
prod1	−42	25	0	−53	**5**	0	−47	16	0
bc1	3.52343	**6**	2	5.4391	63	2	3.43703	3	2
bienst1	72.9757	56	0	55.5	**19**	0	56.75	21	0
bienst2	88.2326	62	0	73.6667	**35**	1	65.25	20	0
dano3_3	629.604	9	25	576.345⋆	**0**	47	576.396	0	152
dano3_4	646.702	12	26	576.435⋆	**0**	76	576.435⋆	0	156
dano3_5	667.574	16	26	576.994	**0**	106	578.648	0	93
mkc1	−522.815	14	0	−563.1	**7**	0	−595.82	2	0
neos1	85	347	1	68	**258**	1	25	32	0
neos2	898.216	**97**	8	958.977	111	7	–	–	0
neos3	1278.4	**2241**	12	1630.21	2886	8	–	–	0
neos4	−4.81256e+10	1	3	−4.8132e+10	**1**	3	−4.83137e+10	1	1
neos5	−4.81256e+10	1	4	−4.8132e+10	**1**	3	−4.83137e+10	1	1
neos6	137	65	4	93	**12**	12	–	–	4
neos7	5.2039e+06	621	0	1.10593e+06	**53**	1	–	–	0
nug08	214⋆	**0**	2	214⋆	**0**	3	214⋆	0	5
qap10	442	30	27	386	**14**	62	366	8	71
seymour1	435.9	6	2	427.063	**4**	3	412.631	0	11
swath1	499.711	32	1	439.106	**16**	2	–	–	0
swath2	1337.12	247	0	641.544	**67**	2	–	–	0
acc-0	0⋆	**0**	0	0⋆	**0**	0	0⋆	0	0
acc-1	0⋆	**0**	2	0⋆	**0**	1	0⋆	0	1
acc-2	0⋆	**0**	2	0⋆	**0**	3	–	–	4
acc-3	0⋆	**0**	10	0⋆	**0**	12	0⋆	0	4
acc-4	–	–	3600	–	–	3600	–	–	8
acc-5	0⋆	**0**	1847	0⋆	**0**	2054	–	–	5
acc-6	0⋆	**0**	1017	–	–	3600	–	–	5
ic97_potential	4568	15	2	4433	**11**	2	–	–	0
ic97_tension	4487	**14**	1	4539	15	1	–	–	0
icir97_tension	7309	14	8	7288	**13**	9	–	–	0
icir97_potential	7724	17	62	7526	**14**	24	–	–	1
nh97_potential	1598	13	10	1554	**10**	17	–	–	0
nh97_tension	1575	4	10	1511⋆	**0**	11	–	–	0
B10-011000	117462	499	1	108472	**453**	1	31263	59	1
B10-011001	117109	444	0	108472	**404**	1	47658	122	1
B11-010000	219275	547	2	215163	**535**	1	89207	163	4
B11-110001	206342	**342**	3	208823	347	4	78142	67	13

(*continued on next page*)

Table 1 (*continued*)

Name	Original feasibility pump			Objective feasibility pump			CPLEX 9.03		
	Objective	Gap %	Time	Objective	Gap %	Time	Objective	Gap %	Time
B12-111111	80931	**89**	46	83096	94	35	–	–	14
C10-001000	255030	2124	0	159137	**1288**	0	23408	104	0
C10-100001	239789	1135	1	134440	**593**	1	–	–	2
C11-010100	146524	450	1	136976	**415**	2	90640	241	2
C11-011100	130241	488	1	128149	**479**	1	32424	46	1
C12-100000	534256	1283	4	456140	**1080**	5	109230	183	19
C12-111100	115593	209	2	113210	**203**	2	110131	195	4
neos10	2	**100**	41	2	**100**	5	−142	87	18
neos16	458	2	88	451	**0**	383	–	–	0
neos20	−199	**54**	3	−199	**54**	7	–	–	0
Geom. mean (116)		43.9	3.2		23.3	3.8		–	2.4
Geom. mean (77)		49.5	1.8		25.9	2.1		11.0	2.0

gap to the optimal or best-known solution was reduced from 43.9% to 23.3%, while the running time increased only slightly.

The comparison to CPLEX yields that the feasibility pump in both versions performs very well in finding an initial solution: the original version only fails on four, the modified version on five instances, while CPLEX cannot find a solution in the root node for 43 instances. The objective feasibility pump, however, produced solutions that are much closer to the ones of CPLEX w.r.t. quality. It could even find the optimal or best-known solution in 12 cases, whereas this was achieved only nine times by the original feasibility pump and only eight times by CPLEX. These two properties – the ability to find solutions and their quite good quality – show that the objective feasibility pump is indeed a reasonable root node heuristic.

As noted earlier, the results of CPLEX include the application of improvement heuristics such as RINS. Disabling RINS, however, produced only slightly worse solutions in nearly the same time: on the 77 instances where all three algorithms found a solution the geometric mean of the gap was 13.1% while it was still 2.0 s for the time. A comparison to CPLEX's *first* solution found in the root node as suggested by Fischetti, Glover, and Lodi [8] resulted in 20.7% gap and 1.5 s.

The different behaviour of the two feasibility pump versions is displayed in Fig. 1 for selected problem instances. The figures show the evolution of the objective values of the LP solutions $x^\star$ and their fractionalities $f^I(x^\star)$ during the course of the algorithm. The graphs on the left-hand side arise from the unmodified version, while the ones on the right-hand side arise from the objective feasibility pump.

In the upper plots (`aflow40b`) one can see that the original version of the feasibility pump rapidly left the region of small objective values while the fractionality measure decreased quite fast. However, the algorithm was not able to drive the solution to integrality before restarting at iteration 10. At this point, the objective value was already far away from the optimal solution value of 1168. In contrast, the objective feasibility pump stayed much closer to the optimal solution but did not decrease the fractionality measure as fast. Nevertheless, after 15 iterations a feasible solution was found with an objective value that was already exceeded after four iterations of the unmodified feasibility pump.

The two plots of `rococoB11-010000` show an example where both versions produced a similar solution in the same number of iterations, although the two algorithms behaved differently. Again, the unmodified feasibility pump increased the objective value and decreased the fractionality faster than the objective feasibility pump.

The bottom plots (`rout`) show a situation where the objective feasibility pump is inferior. The original version performed 35 restarts, most of which can be seen as spikes in the fractionality graph. The last random perturbation at iteration 96 'coincidentally' produced a feasible solution. The objective feasibility pump did not succeed to drive the fractionality to a value less than 0.5 until the last two iterations. Only six restarts were performed. Interestingly, the cycle between two points from iteration 12–30 was left without a restart just by decreasing α.

As already shown by Fischetti, Glover, and Lodi [8] and Bertacco, Fischetti, and Lodi [6] the feasibility pump is a useful heuristic for mixed integer programming, because it usually finds feasible solutions in a reasonable amount of time. Our results show that the modification presented in this paper further improves the feasibility pump: the quality of the resulting solutions is substantially enhanced with only a slight increase in the running time.

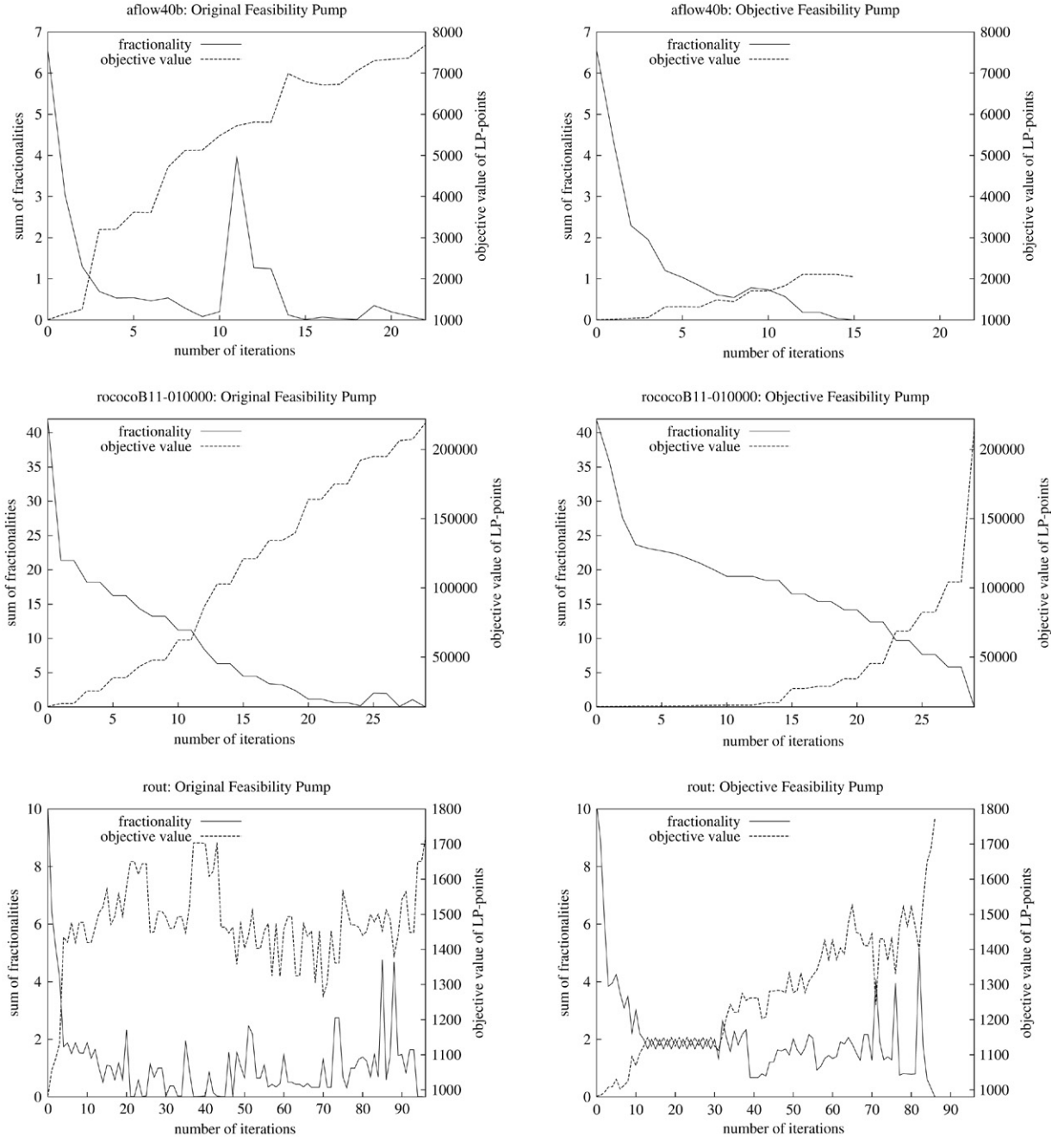

Fig. 1. Evolution of the objective value and fractionality for `aflow40b`, `rococoB11-010000`, and `rout`.

Acknowledgments

We thank Livio Bertacco, Matteo Fischetti, and Andrea Lodi for providing the source code of their original feasibility pump.

References

[1] T. Achterberg, T. Koch, A. Martin, The mixed integer programming library: Miplib 2003. http://miplib.zib.de.
[2] T. Achterberg, T. Koch, A. Martin, Miplib 2003, Operations Research Letters 34 (4) (2006) 1–12.

[3] E. Balas, S. Ceria, M. Dawande, F. Margot, G. Pataki, OCTANE: A new heuristic for pure 0-1 programs, Operations Research 49 (2) (2001) 207–225.
[4] E. Balas, C.H. Martin, Pivot-and-complement: A heuristic for 0-1 programming, Management Science 26 (1) (1980) 86–96.
[5] E. Balas, S. Schmieta, C. Wallace, Pivot and shift—a mixed integer programming heuristic, Discrete Optimization 1 (1) (2004) 3–12.
[6] L. Bertacco, M. Fischetti, A. Lodi, A feasibility pump heuristic for general mixed-integer problems, Technical Report OR/05/5, DEIS – Università di Bologna, Italy, May 2005.
[7] E. Danna, E. Rothberg, C. Le Pape, Exploring relaxation induced neighborhoods to improve MIP solutions, Mathematical Programming 102 (1) (2005) 71–90.
[8] M. Fischetti, F. Glover, A. Lodi, The feasibility pump, Mathematical Programming 104 (1) (2005) 91–104.
[9] M. Fischetti, A. Lodi, Local branching, Mathematical Programming 98 (1–3) (2003) 23–47.
[10] F. Glover, M. Laguna, General purpose heuristics for integer programming — part I, Journal of Heuristics 3 (1997).
[11] F. Glover, M. Laguna, General purpose heuristics for integer programming — part II, Journal of Heuristics 3 (1997).
[12] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publisher, Boston, Dordrecht, London, 1997.
[13] F. Glover, A. Løkketangen, D.L. Woodruff, Scatter search to generate diverse MIP solutions, in: M. Laguna, J. González-Velarde (Eds.), OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, Kluwer Academic Publishers, 2000, pp. 299–317.
[14] F.S. Hillier, Efficient heuristic procedures for integer linear programming with an interior, Operations Research 17 (1969) 600–637.
[15] ILOG, Cplex. http://www.ilog.com/products/cplex.
[16] A. Løkketangen, F. Glover, Solving zero/one mixed integer programming problems using tabu search, European Journal of Operations Research 106 (1998) 624–658.
[17] H. Mittelmann, Decision tree for optimization software: Benchmarks for optimization software. http://plato.asu.edu/bench.html, 2003.
[18] M. Nediak, J. Eckstein, Pivot, cut, and dive: A heuristic for 0-1 mixed integer programming, Technical Report RRR 53-2001, Rutgers University, 2001.
[19] R.M. Saltzman, F.S. Hillier, A heuristic ceiling point algorithm for general integer linear programming, Management Science 38 (2) (1992) 263–283.