The feasibility pump

Matteo Fischetti[*], Fred Glover[+], Andrea Lodi[°]

[*]DEI, University of Padova, matteo.fischetti@unipd.it

[+]Leeds School of Business, University of Colorado at Boulder, Fred.Glover@colorado.edu

[°]DEIS, University of Bologna, alodi@deis.unibo.it

Preliminary draft, November 2003; revised May 2004

Abstract

In this paper we consider the NP-hard problem of finding a feasible solution (if any exists) for a generic MIP problem of the form $\min\{c^T x : Ax \geq b, x_j \text{ integer } \forall j \in \mathcal{I}\}$. Trivially, a feasible solution can be defined as a point $x^* \in P := \{x : Ax \geq b\}$ that is equal to its rounding $\widetilde{x}$, where the rounded point $\widetilde{x}$ is defined by $\widetilde{x}_j := [x^*_j]$ if $j \in \mathcal{I}$ and $\widetilde{x}_j := x^*_j$ otherwise, and $[\cdot]$ represents scalar rounding to the nearest integer. Replacing "equal" with "as close as possible" relative to a suitable distance function $\Delta(x^*, \widetilde{x})$, suggests the following *Feasibility Pump* (`FP`) heuristic for finding a feasible solution of a given MIP.

We start from any $x^* \in P$, and define its rounding $\widetilde{x}$. At each `FP` iteration we look for a point $x^* \in P$ that is as close as possible to the current $\widetilde{x}$ by solving the problem $\min\{\Delta(x, \widetilde{x}) : x \in P\}$. Assuming $\Delta(x, \widetilde{x})$ is chosen appropriately, this is an easily solvable LP problem. If $\Delta(x^*, \widetilde{x}) = 0$, then x^* is a feasible MIP solution and we are done. Otherwise, we replace $\widetilde{x}$ by the rounding of x^*, and repeat.

From a geometric point of view, the `FP` generates two trajectories of points x^* and $\widetilde{x}$ that satisfy feasibility in a complementary but partial way—one satisfies the linear constraints, the other the integer requirement. The `FP` can also be viewed as a strategy for making a heuristic sequence of roundings that yields a feasible MIP point.

We report computational results on a set of 83 difficult 0-1 MIPs, using the commercial software `ILOG-Cplex` 8.1 as a benchmark. The outcome is that `FP`, in spite of its simple foundation, proves competitive with `ILOG-Cplex` both in terms of speed and quality of the first solution delivered. Interestingly, `ILOG-Cplex` could not find any feasible solution at the root node for 19 problems in our test-bed, whereas `FP` was unsuccessful in just 3 cases.

1 Introduction

In this paper we address the problem of finding a feasible solution of a generic MIP problem of the form

$$(MIP) \qquad \min c^T x \tag{1}$$
$$Ax \geq b \tag{2}$$
$$x_j \text{ integer} \quad \forall j \in \mathcal{I} \tag{3}$$

where A is an $m \times n$ matrix. This NP-hard problem can be extremely hard in practice—in some important practical cases, state-of-the-art MIP solvers may spend a very large computational effort before discovering their first solution. Therefore, heuristic methods to find a feasible solution for hard MIPs are highly important in practice. This is particularly true in recent years where successful local-search approaches for general MIPs such as local branching [8] and RINS/guided dives [6] are used that can only be applied if an initial feasible solution is known. Heuristic approaches to general MIP problems have been proposed by several authors, including [1, 2, 3, 6, 8, 9, 10, 11, 12, 13, 17, 16, 20, 21, 24].

Let $P := \{x : Ax \geq b\}$ denote the polyhedron associated with the LP relaxation of the given MIP. With a little abuse of notation, we say that a point x is *integer* if x_j is integer for all $j \in \mathcal{I}$ (no matter the value of the other components). Analogously, the rounding $\widetilde{x}$ of a given x is obtained by setting $\widetilde{x}_j := [x_j]$ if $j \in \mathcal{I}$ and $\widetilde{x}_j := x_j$ otherwise, where $[\cdot]$ represents scalar rounding to the nearest integer.

Throughout this paper we will consider the L_1-norm distance between a generic point $x \in P$ and a given integer $\widetilde{x}$, defined as

$$\Delta(x, \widetilde{x}) = \sum_{j \in \mathcal{I}} |x_j - \widetilde{x}_j|$$

Notice that the continuous variables x_j ($j \notin \mathcal{I}$), if any, do not contribute to this function. Assuming without loss of generality that the MIP constraints include the variable bounds $l_j \leq x_j \leq u_j$ for all $j \in \mathcal{I}$, we can write

$$\Delta(x, \widetilde{x}) := \sum_{j \in \mathcal{I}: \widetilde{x}_j = l_j} (x_j - l_j) + \sum_{j \in \mathcal{I}: \widetilde{x}_j = u_j} (u_j - x_j) + \sum_{j \in \mathcal{I}: l_j < \widetilde{x}_j < u_j} (x_j^+ + x_j^-)$$

where the additional variables x_j^+ and x_j^- require the introduction into the MIP model of the additional constraints:

$$x_j = \widetilde{x}_j + x_j^+ - x_j^-, \quad x_j^+ \geq 0, \; x_j^- \geq 0, \quad \forall j \in \mathcal{I} : l_j < \widetilde{x}_j < u_j \tag{4}$$

Given an integer $\widetilde{x}$, the closest point $x^* \in P$ can therefore be determined by solving the LP

$$\min\{\Delta(x, \widetilde{x}) : Ax \geq b, \; (4)\} \tag{5}$$

If $\Delta(x^*, \widetilde{x}) = 0$, then $x_j^* (= \widetilde{x}_j)$ is integer for all $j \in \mathcal{I}$, so x^* (but not necessarily $\widetilde{x}$) is a feasible MIP solution. Conversely, given a point $x^* \in P$, the integer point $\widetilde{x}$ closest to x^*

is easily determined by rounding x^*. These observations suggest the following *Feasibility Pump* (FP) heuristic to find a feasible MIP solution, in which a pair of points $(x^*, \widetilde{x})$ with $x^* \in P$ and $\widetilde{x}$ integer is iteratively updated with the aim of reducing as much as possible their distance $\Delta(x^*, \widetilde{x})$.

We start from any $x^* \in P$, and initialize a (typically infeasible) integer $\widetilde{x}$ as the rounding of x^*. At each FP iteration, called a *pumping cycle*, we fix $\widetilde{x}$ and find through linear programming the point $x^* \in P$ which is as close as possible to $\widetilde{x}$. If $\Delta(x^*, \widetilde{x}) = 0$, then x^* is a MIP feasible solution, and we are done. Otherwise, we replace $\widetilde{x}$ by the rounding of x^* so as to further reduce $\Delta(x^*, \widetilde{x})$, and repeat. (This basic scheme will be slightly elaborated in the next sections, so as to overcome possible stalling and cycling issues.)

From a geometric point of view, the FP generates two (hopefully convergent) trajectories of points x^* and $\widetilde{x}$ that satisfy feasibility in a complementary but partial way—one satisfies the linear constraints, the other the integer requirement. An important feature of the method is related to the infeasibility measure used to guide $\widetilde{x}$ towards feasibility: instead of taking a weighted combination of the degree of violation of the single linear constraints, as customary in MIP heuristics, we use the distance $\Delta(x^*, \widetilde{x})$ of $\widetilde{x}$ from polyhedron P, as computed at each pumping cycle[1]. This distance can be interpreted as a sort of "difference of pressure" between the two complementary types of infeasibility of x^* and $\widetilde{x}$, that we try to reduce by "pumping" the integrality of $\widetilde{x}$ into x^*—hence the name of the method. FP can be interpreted as a strategy for producing a sequence of roundings that leads to a feasible MIP point.

The FP can also be viewed as modified *local branching* strategy [8]. Indeed, at each pumping cycle we have an incumbent (infeasible) solution $\widetilde{x}$ satisfying the integer requirement, and we face the problem of finding a feasible solution (if any exists) within a small-distance neighborhood, i.e., changing only a small subset of its variables. In the local branching context, this subproblem would have been modeled by the MIP

$$\min\{c^T x : Ax \geq b, x_j \text{ integer } \forall j \in \mathcal{I}, (4), \, \Delta(x, \widetilde{x}) \leq k\}$$

for a suitable value of parameter k, and solved through an enumerative MIP method. In the FP context, instead, the same subproblem is modeled in a relaxed way through the LP (5), where the "small distance" requirement is translated in terms of the objective function. (Notice that (5) can be viewed as a relaxed model for the problem: "Change a minimum number of variables so as to convert the current $\widetilde{x}$ into a feasible MIP solution x^*".) The working hypothesis here is that the objective function $\Delta(x, \widetilde{x})$ will discourage the optimal solution x^* of the relaxation from being "too far" from the incumbent $\widetilde{x}$, hence we expect a large number of the integer-constrained variables in $\widetilde{x}$ will retain their (integer) values also in the optimal x^*.

In spite of its simple nature, FP proved quite effective in finding feasible solutions of hard MIPs, though sometimes it stops prematurely due to stalling issues. This happens in the first iteration where $\Delta(x^*, \widetilde{x})$ is not reduced when replacing $\widetilde{x}$ by the rounding of x^*, meaning that all the integer-constrained components of $\widetilde{x}$ would stay unchanged in this iteration. In this case, if $\Delta(x^*, \widetilde{x}) > 0$ we still want to modify $\widetilde{x}$, even if this results in an increased

[1] A similar infeasibility measure for nonlinear problems was recently investigated in [5].

distance from x^*. This can be obtained by increasing (respectively, decreasing) to a next integer value just a few components $\tilde{x}_j$ with $j \in \mathcal{I}$ and $x_j^* > \tilde{x}_j$ (resp., $x_j^* < \tilde{x}_j$), chosen so as to minimize the increase in the current value of $\Delta(x^*, \tilde{x})$.

The paper is organized as follows. In Section 2 we present the `FP` method in more details, focusing on its implementation for 0-1 MIPs. The possibility of reducing the computing time involved in the various LP solutions is addressed in Section 3, where the use of approximate LP solutions is investigated. In the same section we also address the possibility of using the `FP` scheme to produce a sequence of feasible solutions of better and better quality. Some conclusions are finally drawn in Section 4.

Throughout the paper, we report computational results comparing the performance of the proposed `FP` method with that of the commercial software `ILOG-Cplex` 8.1. Our testbed is made by 44 0-1 MIP instances collected in MIPLIB[2] 2003 [19] and described in Table 1, plus an additional set of 39 hard 0-1 MIPs described in Table 2 and available, on request, from the third author. The two tables report the instance names and the corresponding number of variables (n), of 0-1 variables ($|\mathcal{I}|$) and of constraints (m).

Name	n	$\lvert\mathcal{I}\rvert$	m	Name	n	$\lvert\mathcal{I}\rvert$	m
10teams	2025	1800	230	mod011	10958	96	4480
A1C1S1	3648	192	3312	modglob	422	98	291
aflow30a	842	421	479	momentum1	5174	2349	42680
aflow40b	2728	1364	1442	net12	14115	1603	14021
air04	8904	8904	823	nsrand_ipx	6621	6620	735
air05	7195	7195	426	nw04	87482	87482	36
cap6000	6000	6000	2176	opt1217	769	768	64
dano3mip	13873	552	3202	p2756	2756	2756	755
danoint	521	56	664	pk1	86	55	45
ds	67732	67732	656	pp08a	240	64	136
fast0507	63009	63009	507	pp08aCUTS	240	64	246
fiber	1298	1254	363	protfold	1835	1835	2112
fixnet6	878	378	478	qiu	840	48	1192
glass4	322	302	396	rd-rplusc-21	622	457	125899
harp2	2993	2993	112	set1ch	712	240	492
liu	1156	1089	2178	seymour	1372	1372	4944
markshare1	62	50	6	sp97ar	14101	14101	1761
markshare2	74	60	7	swath	6805	6724	884
mas74	151	150	13	t1717	73885	73885	551
mas76	151	150	12	tr12-30	1080	360	750
misc07	260	259	212	van	12481	192	27331
mkc	5325	5323	3411	vpm2	378	168	234

Table 1: The 44 0-1 MIP instances collected in MIPLIB 2003 [19]

[2] Another 0-1 MIP included in the library, namely `stp3d`, was not considered since the computing time required for the first LP relaxation is larger than 1 hour.

Name	n	$\|\mathcal{I}\|$	m	source
biella1	7328	6110	1203	[8]
NSR8K	38356	32040	6284	[8]
dc1c	10039	8380	1649	[7]
dc1l	37297	35638	1653	[7]
dolom1	11612	9720	1803	[7]
siena1	13741	11775	2220	[7]
trento1	7687	6415	1265	[7]
rail507	63019	63009	509	[8]
rail2536c	15293	15284	2539	[8]
rail2586c	13226	13215	2589	[8]
rail4284c	21714	21705	4284	[8]
rail4872c	24656	24645	4875	[8]
A2C1S1	3648	192	3312	[8]
B1C1S1	3872	288	3904	[8]
B2C1S1	3872	288	3904	[8]
sp97ic	12497	12497	1033	[8]
sp98ar	15085	15085	1435	[8]
sp98ic	10894	10894	825	[8]
bg512142	792	240	1307	[18]
dg012142	2080	640	6310	[18]

Name	n	$\|\mathcal{I}\|$	m	source
blp-ar98	16021	15806	1128	[17]
blp-ic97	9845	9753	923	[17]
blp-ic98	13640	13550	717	[17]
blp-ir98	6097	6031	486	[17]
CMS750_4	11697	7196	16381	[14]
berlin_5_8_0	1083	794	1532	[14]
railway_8_1_0	1796	1177	2527	[14]
usAbbrv.8.25_70	2312	1681	3291	[14]
manpower1	10565	10564	25199	[22]
manpower2	10009	10008	23881	[22]
manpower3	10009	10008	23915	[22]
manpower3a	10009	10008	23865	[22]
manpower4	10009	10008	23914	[22]
manpower4a	10009	10008	23866	[22]
ljb2	771	681	1482	[6]
ljb7	4163	3920	8133	[6]
ljb9	4721	4460	9231	[6]
ljb10	5496	5196	10742	[6]
ljb12	4913	4633	9596	[6]

Table 2: The additional set of 39 0-1 MIP instances

The outcome of our experiments can be summarized as follows. `FP`, even in its basic version, compares favorably with the `ILOG-Cplex` heuristics applied at the root node: though `FP` and `ILOG-Cplex` (root node) require a comparable computing time, the percentage of success in finding a feasible solution is 96.3% for `FP`, and 77.1% for `ILOG-Cplex` (in its best-tuned version). In this respect, the `FP` performance is very satisfactory: whereas `ILOG-Cplex` could not find any feasible solution at the root node in 19 cases (and in 4 cases even allowing for 1,800 seconds of branching), `FP` was unsuccessful only 3 times. Also interesting is the comparison of the quality of the `FP` solution with that found by the root-node `ILOG-Cplex` heuristics: the latter delivered a strictly-better solution in 33 cases, whereas the solution found by `FP` was strictly better in 46 cases. The computing times to get to the first feasible solution appear comparable: excluding the instances for which both methods required less than 1 second, `ILOG-Cplex` was faster in 26 cases, and `FP` was faster in 31 cases.

A preliminary version of the present paper was presented at the Combinatorial Optimization meeting held in Aussois, January 4-10, 2004.

2 The basic feasibility pump

As already mentioned, in the sequel we will always consider 0-1 MIPs in which all integer-constrained variables are binary, i.e., we assume constraints $Ax \geq b$ include the variable bounds $0 \leq x_j \leq 1$ for all $j \in \mathcal{I}$. As a consequence, no additional variables x_j^+ and x_j^- are

The Feasibility Pump (basic version):

1. initialize nIT := 0 and $x^* := \text{argmin}\{c^T x : Ax \geq b\}$;
2. if x^* is integer, return(x^*);
3. let $\widetilde{x} := [x^*]$ (= rounding of x^*);
4. while (time < TL) do
5.   let nIT := nIT $+1$ and compute $x^* := \text{argmin}\{\Delta(x, \widetilde{x}) : Ax \geq b\}$;
6.   if x^* is integer, return(x^*);
7.   if $\exists\, j \in \mathcal{I} : [x_j^*] \neq \widetilde{x}_j$ then
8. $\widetilde{x} := [x^*]$

       else

9.   flip the TT = rand(T/2,3T/2) entries $\widetilde{x}_j$ $(j \in \mathcal{I})$ with highest $|x_j^* - \widetilde{x}_j|$
10.   endif
11. enddo

Figure 1: The basic FP implementation

required in the definition of the distance function (4), which attains the simpler form

$$\Delta(x, \widetilde{x}) := \sum_{j \in \mathcal{I} : \widetilde{x}_j = 0} x_j + \sum_{j \in \mathcal{I} : \widetilde{x}_j = 1} (1 - x_j) \tag{6}$$

An outline of the FP algorithm for 0-1 MIPs is reported in Figure 1. The algorithm receives on input two parameters: the time limit TL and the number T of variables to flip each iteration—the use of this latter parameter will be clarified later on.

At step 1, x^* is initialized as a minimum-cost solution of the LP relaxation, a choice intended to increase the chance of finding a small-cost feasible solution. At each pumping cycle, at step 5 we redefine x^* as a point in P with minimum distance from the current integer $\widetilde{x}$. We then check whether the new $x^* \in P$ is integer. If this is not the case, the current integer point $\widetilde{x}$ is replaced at step 8 by $[x^*]$, so as to reduce even further the current distance $\Delta(x^*, \widetilde{x})$. In order to avoid stalling issues, in case $\widetilde{x} = [x^*]$ (with respect to the integer-constrained components) we flip, at step 9, a random number TT $\in \{\frac{1}{2}\mathtt{T}, \cdots, \frac{3}{2}\mathtt{T}\}$ of integer-constrained entries of $\widetilde{x}$, chosen so as to minimize the increase in the total distance $\Delta(x^*, \widetilde{x})$.

The procedure terminates as soon as a feasible integer solution x^* is found, or when the time-limit TL has been exceeded. In this latter case, the FP heuristic has to report a failure–which is not surprising, as finding a feasible 0-1 MIP solution is an NP-hard problem in general.

Figure 2 gives an illustration of two sample FP runs with T=20, where the infeasibility measure $\Delta(x^*, \widetilde{x})$ iteratively reduced to zero; note that both sequences are essentially monotone, except for the possibility of small irregularities due to the flips performed at step 9.

A main problem with the basic FP implementation described above is the possibility of *cycling*: after a certain number of iterations, the method may enter a loop where a same sequence of points x^* and $\tilde{x}$ is visited again and again. In order to overcome this drawback, we implemented the following straightforward perturbation mechanism. As soon as a cycle is heuristically detected by comparing the solutions found in the last 3 iterations, and in any case after `R` (say) iterations, we skip steps 7-10 and apply a random perturbation move. To be more specific, for each $j \in \mathcal{I}$ we generate a uniformly random value $\rho_j \in [-0.3, 0.7]$ and flip $\tilde{x}_j$ in case $|x_j^* - \tilde{x}_j| + \max\{\rho_j, 0\} > 0.5$.

The results of the initial `FP` implementation described above are reported in Tables 3 and 4, with a comparison with the state-of–the-art MIP solver `ILOG-Cplex` 8.1. The focus of this experiment was to measure the capability of the compared methods to converge to an initial feasible solution, hence both `FP` and `ILOG-Cplex` were stopped as soon as the first feasible solution was found. Computing times are expressed in CPU seconds, and refer to a Pentium M 1.6 Ghz notebook with 512 MByte of main memory. Parameters `T` and `TL` were set to 20 and 1,800 CPU seconds, respectively, while the perturbation-frequency parameter `R` was set to 100.

In the `FP` implementation, we use the `ILOG-Cplex` function `CPXoptimize` to solve each LP (thus leaving to `ILOG-Cplex` the choice of the actual LP algorithm to invoke) with the default parameter setting.

As to `ILOG-Cplex`, after extensive experiments and contacts with `ILOG-Cplex` staff [23] we found that, as far as the time and quality of the root node solution is concerned, the best results are obtained (perhaps surprisingly) when the MIP preprocessing/presolve is not invoked, and the default "balance optimality and integer feasibility" strategy for the exploration of the search tree is used. Indeed, the number of root-node failures for `ILOG-Cplex` was 19 with the setting we used in our experiments. By contrast, when the preprocessing/presolve was activated `ILOG-Cplex` could not find any feasible solution at the root node in 25 cases (with the default "balance optimality and integer feasibility" strategy) or in 41 cases (with the "emphasize integrality" strategy). In case the preprocessing/presolve is deactivated but the "emphasize integrality" strategy was used, instead, no solution was found at the root node in 33 cases.

Tables 3 and 4 report the results for the instances in Tables 1 and 2, respectively. For each instance and for each algorithm (`FP` and `ILOG-Cplex`) we report the value of the first feasible solution found ("value" for `FP`, and "root value/first value" for `ILOG-Cplex`) and the corresponding computing time, in Pentium M-1.6 CPU seconds ("time"). In case of failure, "N/A" is reported. Moreover, for `FP` we report the number of iterations performed by the algorithm ("nIT"), while for `ILOG-Cplex` we give the number of branch-and-bound nodes ("nodes") needed to initialize the incumbent solution.

Our first order of business here was to evaluate the percentage of success in finding a feasible MIP solution without resorting to branching. In this respect, the `FP` performance is very satisfactory: whereas `ILOG-Cplex` could not find any feasible solution at the root node in 19 cases (and in 10 cases even allowing for 1,800 seconds of branching), `FP` was unsuccessful only 3 times.

Also interesting is the comparison of the quality of the `FP` solution with that found by the root-node `ILOG-Cplex` heuristics: the latter delivered a strictly-better solution in 33 cases,

	feasibility pump			ILOG-Cplex 8.1			
name	value	nIT	time	root value	first value	nodes	time
10teams	992.00	53	7.5	N/A	924.00	14	5.2
A1C1S1	18,377.24	5	3.8	N/A	14,264.61	120	8.6
aflow30a	4,545.00	18	0.1	N/A	1,574.00	40	1.4
aflow40b	6,859.00	7	0.5	1,786.00		0	1.8
air04	58,278.00	4	12.5	57,640.00		0	6.2
air05	29,937.00	2	3.4	29,590.00		0	2.0
cap6000	-2,354,320.00	2	0.6	-2,445,344.00		0	0.6
dano3mip	756.62	4	77.7	768.37		0	161.2
danoint	77.00	3	0.2	73.00		0	1.7
ds	N/A	81	1,800.0	5,418.56		0	81.6
fast0507	181.00	4	34.0	209.00		0	33.1
fiber	1,911,617.79	2	0.0	570,936.07		0	0.0
fixnet6	9,131.00	4	0.0	12,163.00		0	0.0
glass4	4,650,037,150.00	23	0.1	N/A	3,500,034,900.00	162	0.3
harp2	-43,856,974.00	654	4.5	-73,296,664.00		0	0.1
liu	6,262.00	0	0.0	6,262.00		0	0.0
markshare1	1,064.00	11	0.0	710.00		0	0.0
markshare2	1,738.00	7	0.0	1,735.00		0	0.0
mas74	52,429,700.59	1	0.0	19,197.47		0	0.0
mas76	194,527,859.06	1	0.0	44,877.42		0	0.0
misc07	4,515.00	123	0.5	3,060.00		0	0.0
mkc	-164.56	2	0.3	-195.97		0	0.5
mod011	-49,370,141.17	0	1.0	-42,902,314.08		0	1.9
modglob	35,147,088.88	0	0.0	20,786,787.02		0	0.0
momentum1	455,740.91	520	1478.4	N/A	N/A	75	1,800.0
net12	337.00	346	55.4	N/A	214.00	480	1,593.7
nsrand_ipx	340,800.00	3	0.7	699,200.00		0	0.3
nw04	19,882.00	1	2.9	17,306.00		0	5.1
opt1217	-12.00	0	0.0	-14.00		0	0.0
p2756	N/A	163435	1,800.0	3,485.00		0	0.1
pk1	57.00	1	0.0	89.00		0	0.0
pp08a	11,150.00	2	0.0	14,800.00		0	0.0
pp08aCUTS	10,940.00	2	0.0	13,540.00		0	0.0
protfold	-10.00	367	493.8	N/A	N/A	637	1,800.0
qiu	389.36	3	0.3	1,691.14		0	0.1
rd-rplusc-21	N/A	900	1,800.0	N/A	N/A	372	1,800.0
set1ch	76,951.50	2	0.0	109,759.00		0	0.0
seymour	452.00	9	3.4	469.00		0	5.1
sp97ar	1,398,705,728.00	6	4.3	734,171,023.04		0	2.6
swath	18,416.00	109	4.7	N/A	826.66	1609	38.6
t1717	826,848.00	42	644.9	N/A	N/A	1397	1,800.0
tr12-30	277,218.00	9	0.1	N/A	143,586.00	200	2.1
van	8.21	4	245.0	6.59		0	100.3
vpm2	19.25	3	0.0	15.25		0	0.0

Table 3: Convergence to a first feasible solution

	feasibility pump			ILOG-Cplex 8.1			
name	value	nIT	time	root value	first value	nodes	time
biella1	3,537,959.54	5	7.9	3,682,135.10		0	8.4
NSR8K	5,111,376,832.18	5	1,751.4	4,923,673,379.32		0	1,478.6
dc1c	27,348,312.19	4	19.3	33,458,468.26		0	15.3
dc1l	8,256,022.49	5	94.4	752,840,672.81		0	67.6
dolom1	298,684,615.17	7	32.1	584,923,856.01		0	29.2
siena1	104,004,996.99	5	91.8	591,385,634.57		0	66.4
trento1	356,179,003.01	2	17.8	621,044,078.07		0	18.1
rail507	178.00	2	41.1	205.00		0	32.9
rail2536c	715.00	4	26.7	771.00		0	27.1
rail2586c	1,007.00	5	81.6	1,072.00		0	68.6
rail4284c	1,124.00	3	1095.8	1,218.00		0	273.1
rail4872c	1,614.00	5	311.9	1,737.00		0	305.6
A2C1S1	19,879.93	5	3.7	20,865.33		0	0.0
B1C1S1	38,530.65	7	5.2	69,933.52		0	0.1
B2C1S1	48,279.95	6	4.5	70,625.52		0	0.1
sp97ic	1,280,793,707.52	3	2.7	515,786,416.96		0	1.7
sp98ar	988,402,511.36	4	4.4	599,527,422.56		0	2.4
sp98ic	959,924,716.00	3	2.1	550,157,878.72		0	1.5
blp-ar98	25,094.03	161	23.6	N/A	9,473.66	50	37.2
blp-ic97	7,874.87	4	0.7	6,408.43		0	0.4
blp-ic98	14,848.96	6	1.4	9,080.53		0	0.6
blp-ir98	5,388.84	3	0.3	2,927.29		0	1.2
CMS750_4	606.00	131	18.9	803.00		0	13.9
berlin_5_8_0	79.00	10	0.1	89.00		0	0.4
railway_8_1_0	440.00	13	0.3	478.00		0	0.4
usAbbrv.8.25_70	164.00	34	0.8	N/A	130.00	6036	46.8
bg512142	120,738,665.00	0	0.1	120,670,203.50		0	0.3
dg012142	153,406,945.50	0	0.8	153,392,273.00		0	1.7
manpower1	8.00	66	38.5	N/A	N/A	34	1,800.0
manpower2	7.00	148	157.9	N/A	N/A	10	1,800.0
manpower3	6.00	49	56.9	N/A	N/A	10	1,800.0
manpower3a	6.00	73	67.4	N/A	N/A	10	1,800.0
manpower4	7.00	192	107.7	N/A	N/A	17	1,800.0
manpower4a	7.00	53	85.1	N/A	N/A	16	1,800.0
ljb2	7.24	0	0.0	1.63		0	0.4
ljb7	8.61	0	0.5	0.81		0	3.9
ljb9	9.48	0	0.8	9.48		0	6.2
ljb10	7.31	0	1.0	7.31		0	6.9
ljb12	6.20	0	0.7	3.21		0	6.4

Table 4: Convergence to a first feasible solution (cont.d)

whereas the solution found by FP was strictly better in 46 cases. The computing times to get to the first feasible solution appear comparable: excluding the instances for which both methods required less than 1 second, ILOG-Cplex was faster in 26 cases, and FP was faster in 31 cases. Finally, column nIT (FP iterations) shows that the number of LPs solved by FP for finding its first feasible solution is typically very small, which confirms the effectiveness of the distance function used at step 5 in driving x^* towards integrality.

Quite surprisingly, sometimes FP requires just a few iterations but takes much more time than expected. E.g., for problem rail4284c in Table 4 the root node of ILOG-Cplex took only 273.1 seconds—including the application of the internal heuristics. FP found a feasible solution after just 3 iterations but the overall computing time was 1095.8 seconds—about 4 times larger. This can be partly explained by observing that FP requires the initial solution of *two* LPs with quite different objective functions: the initialization LP at step 1 (which uses the original objective function), and the LP at the first execution of step 5 (using the distance-related objective function). Hence we take for granted that no effective parametrization between these two LPs can be obtained. However, a better integration of FP with the LP solver is likely to produce improved results in several cases.

As already stated, in our experiments we deliberately avoided any problem-dependent fine tuning of the LP parameters, and used for both FP and ILOG-Cplex their default values. However, some knowledge of the type of instance to be solved can improve both FP and ILOG-Cplex performance considerably, especially for highly degenerate cases. For instance, we found that the choice of the LP algorithm used for re-optimization at step 5 may have a strong impact on the overall FP computing times. E.g., if we force the use of the dual simplex, the overall computing time for rail4284c decreases from 1095.8 to just 311.1 seconds. This is of course true also for ILOG-Cplex. E.g., for manpower instances Bixby [4] suggested an ad-hoc tuning consisting of (a) avoiding the generation of cuts (`set mip cut all -1`), and (b) activating a specific dual-simplex pricing algorithm (`set simp dg 2`). This choice considerably reduces the time spent by the LP solver at each branching node, and allows ILOG-Cplex to find a first feasible solution (of value 6.0) for instances manpower1, manpower2, manpower3, manpower3a, manpower4 and manpower4a after 111, 150, 107, 156, 202 and 197 branching nodes, and after 28.4, 115.4, 99.7, 70.7, 100.2, and 84.7 CPU seconds, respectively.

A pathological case for FP is instance p2756, which can instead be solved very easily by ILOG-Cplex. This is due to the particular structure of this problem, which involves a large number of big-M coefficients. More specifically, several constraints in this model are of the type $\alpha_i^T y \leq \beta_i + M_i z_i$, where M_i is a very large positive value, y is a binary vector, and z_i is a binary variable whose value 1 is used to actually deactivate the constraint. Feasible solutions of this model can obtained quite easily by setting $z_i = 1$ so as to deactivate these constraints. However, this choice turns out to be very expensive in terms of the LP objective function, where variables z_i are associated with large costs. Therefore, the LP solutions (y^*, z^*) tend to associate very small values to all variables z_i^*, namely $z_i^* = \max\{0, (\alpha_i^T y^* - \beta_i)/M_i\}$, which are then systematically rounded down by our scheme. As a consequence, FP is actually looking for a feasible y that fulfills *all* the constraints $\alpha_i^T x \leq \beta_i$—an almost impossible task. This consideration would suggest that a more elaborated `FP` scheme should introduce a mechanism that, in some specific cases, allows some variables to be rounded up no matter their value in the LP solution—a topic that is left to future research.

3 `FP` variants

The basic `FP` scheme will next be elaborated in the attempt of improving (a) the required computing time, and/or (b) the quality of the heuristic solution delivered by the method.

3.1 Reducing the computing time

We have evaluated the following two simple `FP` variants:

1. `FP1`: At step 1, the LP relaxation of the original MIP (i.e., the one with the original objective function $c^T x$) is solved approximately through a primal-dual method (e.g., the `ILOG-Cplex` barrier algorithm), and as soon as a prefixed primal-dual gap γ is reached the solution is stopped and no crossover is performed. The almost-optimal dual variables are then used as Lagrangian multipliers to compute a mathematically-correct lower bound on the optimal LP value. Moreover, at step 5 each LP relaxation is solved approximately via the primal simplex method with a limit of `SIL` simplex pivots (if this limit is reached within the simplex phase 1, the approximate LP solution x^* is not guaranteed to be primal feasible, hence we skip step 6).

2. `FP2`: The same as `FP1`, but at step 1 the first $\widetilde{x}$ is obtained by just rounding a random initial solution $x^* \in [0,1]^n$ (no LP solution is required).

3.2 Improving the solution quality

As stated, the `FP` method is designed to provide a feasible solution to hard MIPs—no particular attention is paid to the *quality* of this solution. In fact, the original MIP objective function is only used for the initialization of $\widetilde{x}$ in step 1—while it is completely ignored in variant `FP2` above. On the other hand, `FP` proved quite fast in practice, and one may think of simple modifications to provide a *sequence* of feasible solutions of better and better quality.[3] We have therefore investigated a natural extension of our method, based on the idea of adding the upper-bound constraint $c^T x \leq UB$ to the LPs solved at step 5, where UB is updated dynamically each time a new feasible solution is found. To be more specific, right after step 1 we initialize $z^*_{LP} = c^T x^*$ (= LP relaxation value) and $UB = +\infty$. Each time a new feasible solution x^* of value $z^H = c^T x^*$ is found at step 5, we update $UB = \alpha z^*_{LP} + (1-\alpha) z^H$ for $\alpha \in (0,1)$, and continue the while-do loop. Furthermore, in the

[3] A possible way to improve the quality of the first solution found by `FP` is of course to exploit local-search methods based on enumeration of a suitable solution neighborhood of the first feasible solution found, such as the recently-proposed local branching [8], RINS or guided dives [6] schemes.

test at step 4 we add the condition `nIT-nIT0 < IL`, where `nIT0` gives the value of `nIT` when the first feasible solution is found (`nIT0`=0 if none is available), and the input parameter `IL` gives the maximum number of additional `FP` iterations allowed after the initialization of the incumbent solution.

The above scheme can also be applied to variant `FP1`, where the LP at step 1 is solved approximately. As to `FP2`, where no bound is computed, z^*_{LP} is left undefined and the upper bound UB is heuristically reduced after each solution updating as $UB = z^H - \beta|z^H|$ (assuming $z^H \neq 0$).

A final comment is in order. Due to the additional constraint $c^T x \leq UB$, it is often the case that the integer components of $\widetilde{x}$ computed at step 8 define a feasible point for the *original* system $Ax \geq b$, but not for the current one. In order to improve the chances of updating the incumbent solution, right after step 8 we therefore apply a simple post-processing of $\widetilde{x}$, consisting in solving the LP $\min\{c^T x : Ax \geq b, \ x_j = \widetilde{x}_j \ \forall j \in \mathcal{I}\}$ and comparing the corresponding solution (if any exists) with the incumbent one.

3.3 Computational experiments

Table 5 reports the results of the feasibility pump variants `FP1` and `FP2`. For this experiment we selected 26 instances out of the 83 in our testbed, chosen as those for which (a) both `FP` and `ILOG-Cplex` were able to find a solution within the time limit of 1,800 CPU seconds, and (b) the computing time required by either `ILOG-Cplex` or `FP` was at least 10 CPU seconds. We also included the `manpower` instances, and ran `ILOG-Cplex` with the ad-hoc tuning described in the previous section.

For these reduced testbed, we evaluated the capability of `FP1` and `FP2` to converge quickly to an initial solution (even if worse than that produced by `FP`) and to improve it in a given amount of additional iterations. The underlying idea is that, for problems in which the LP solution is very time consuming, it may be better to solve the LPs approximately, while trying to improve the first (possibly poor) solutions at a later time.

For the experiments reported in Table 5 the parameters were set as follows: $\alpha = 0.50$, $\beta = 0.25$, $\gamma = 0.20$, $\texttt{SIL} = 1,000$, and $\texttt{IL} = 250$.

In the table, the `ILOG-Cplex` columns are taken from the previous experiments. For both `FP1` and `FP2` we report the time and value of the first solution found, and the time and value of the best solution found after `IL`=250 additional `FP` iterations. Moreover, for `FP1` we report the extra computing time spent for computing the initial lower bound through the (approximate) `ILOG-Cplex` barrier method ("LB time").

According to the table, `FP2` is able to deliver its first feasible solution within an extremely short computing time—often 1-2 orders of magnitude shorter than `ILOG-Cplex` and `FP`. E.g., `FP2` took only 1.5 seconds for `NSR8K`, whereas `ILOG-Cplex` and `FP` required 1,478.6 and 1,751.4 seconds, respectively. In three cases however the method did not find any solution within the 1,800-second time limit. The quality of the first solution is of course poor (remember that the MIP objective function is completely disregarded until the first feasible solution is found), but it improves considerably during subsequent iterations. At the end of its execution, `FP2` was faster than ILOG-Cplex in 12 out of the 26 cases, and returned a better (or equal) solution in 11 cases.

FP1 performs somewhat better than this. Its first solution is much better than that of FP2 and strictly better than the ILOG-Cplex solution in 4 cases; the corresponding computing time (increased by the LB time) is shorter than that of ILOG-Cplex in 22 out of the 26 cases. After 250 more FP iterations, the quality of the FP1 solution is equal to that of ILOG-Cplex in 6 cases, strictly better in 12 cases, and worse in 8 cases; the corresponding computing time compares favorably with that of ILOG-Cplex in 12 cases.

4 Conclusions

We have proposed and analyzed computationally a new heuristic method for finding a feasible solution to general MIP problems. The approach, called the Feasibility Pump (FP), generates two trajectories of points x^* and $\tilde{x}$ that satisfy MIP feasibility in a complementary but partial way—one satisfies the linear constraints, the other the integer requirement. The method can also be interpreted as a strategy for making a heuristic sequence of roundings that yields a feasible MIP point.

We report computational results on a set of 83 difficult 0-1 MIPs, using the commercial software ILOG-Cplex 8.1 as a benchmark. The outcome is that FP is competitive with ILOG-Cplex both in terms of speed and quality of the first solution delivered. Interestingly, ILOG-Cplex could not find any feasible solution at the root node for 19 problems in our test-bed, whereas FP was unsuccessful in just 3 cases.

Future directions of research should address the topic of better exploiting the considerable amount of information provided by the FP method. Indeed, even in case of failure, the infeasible point $x^* \in P$ with minimum distance from its rounding (chosen among those generated by the FP procedure) is likely to be well suited to start a "feasibility recovery" procedure based on *enumerative* local-search methods in the spirit of local branching [8], or RINS/guided dives [6].

5 Acknowledgements

The work of the first and last authors was supported by MIUR and CNR, Italy, and by the EU project ADONET. The work of the second author was supported by the Center for Disease Control of the U.S. National Center for Health Statistics. The authors are grateful to Dimitris Bertsimas for interesting discussions on the role of randomness in rounding.

	ILOG-Cplex 8.1		FP2: no bound, random initial solution					FP1: approximate solution of LPs					
name	first value	time	first value	nIT	time	best value	time	LB time	first value	nIT	time	best value	time
air04	57,640.00	6.2	N/A	5210	1,800	N/A	1,800	1.3	62,398.00	599	441.8	59,807.00	973.8
dano3mip	768.37	161.2	N/A	5531	1,800	N/A	1,800	12.1	2,649,999.80	241	64.2	155,597.13	143.1
fast0507	209.00	33.1	60,770.00	1	0.6	198.00	63.7	4.2	205.00	2	1.2	188.00	25.7
net12	214.00	1593.7	337.00	1259	116.4	337.00	136.7	42.9	337.00	63	6.4	337.00	26.3
swath	826.66	38.6	46,277.73	39	2.1	1,512.21	17.1	0.3	45,023.47	382	15.8	45,023.47	17.1
van	6.59	100.3	N/A	517	1,800	N/A	1,800	72.5	22.75	159	649.7	22.75	1,730.2
NSR8K	4,923,673,379.32	1478.6	3,431,645,501.71	2	1.5	279,901,658.55	108.4	218.4	3,568,380,063.65	3	2.0	268,661,660.39	340.5
dc1c	33,458,468.26	15.3	195,977,054.50	4	0.8	10,732,532.92	87.3	12.6	8,644,498,480.28	2	0.4	5,074,719.02	97.8
dc1l	752,840,672.81	67.6	41,577,097,275.19	0	0.3	27,865,094.81	167.2	11.7	126,035,913.11	2	1.7	11,498,038.84	172.4
dolom1	584,923,856.01	29.2	431,992,801.00	28	13.3	149,854,956.11	116.3	12.4	475,952,465.07	24	10.8	155,077,538.15	130.3
siena1	591,385,634.57	66.4	8,883,564,918.89	1	0.6	139,122,554.83	360.9	44.6	953,570,679.98	3	1.3	430,116,204.90	340.5
trento1	621,044,078.07	18.1	1,296,470,184.01	15	3.0	65,746,910.00	137.4	8.4	1,296,470,184.01	15	3.0	86,011,231.01	38.9
rail507	205.00	32.9	20,251.00	1	0.8	220.00	41.9	5.2	247.00	2	1.8	187.00	89.9
rail2536c	771.00	27.1	2,430.00	1	0.3	717.00	553.3	14.5	919.00	1	0.3	718.00	450.1
rail2586c	1,072.00	68.6	2,900.00	1	0.2	1,122.00	134.8	5.1	1,376.00	1	0.3	1,028.00	735.4
rail4284c	1,218.00	273.1	4,531.00	1	0.5	2,067.00	113.3	50.7	1,554.00	2	0.8	1,174.00	121.2
rail4872c	1,737.00	305.6	4,513.00	2	0.8	3,385.00	108.1	17.7	2,132.00	2	1.1	1,611.00	197.7
blp-ar98	9,473.66	37.2	25,459.18	562	64.5	25,459.18	84.5	1.6	24,876.87	959	106.7	24,876.87	111.5
CMS750_4	803.00	13.9	1,000.00	4	2.9	748.00	34.9	1.0	1,000.00	3	1.9	742.00	33.1
usAbbrv.8.25_70	130.00	46.8	195.00	3	0.1	195.00	4.8	0.1	189.00	8	0.2	180.00	4.1
manpower1	6.00*	28.4	9.00	13	3.3	7.00	12.7	38.8	12.00	21	4.5	6.00	14.1
manpower2	6.00*	115.4	8.00	53	11.5	6.00	20.5	55.6	8.00	24	6.0	6.00	14.7
manpower3	6.00*	99.7	7.00	21	5.1	7.00	13.7	50.4	11.00	85	17.4	6.00	26.3
manpower3a	6.00*	70.7	10.00	120	25.9	6.00	35.0	52.6	9.00	64	14.7	6.00	23.6
manpower4	6.00*	100.2	6.00	169	36.6	6.00	45.1	49.6	10.00	43	9.4	6.00	18.0
manpower4a	6.00*	84.7	7.00	24	6.3	7.00	14.7	52.1	9.00	21	6.0	6.00	14.7

Table 5: Performance of two FP variants (* ILOG-Cplex was run with an ad-hoc tuning)

References

[1] E. Balas, S. Ceria, M. Dawande, F. Margot, G. Pataki. OCTANE: A New Heuristic For Pure 0-1 Programs. *Operations Research* 49, 207–225, 2001.

[2] E. Balas and C.H. Martin. Pivot-And-Complement: A Heuristic For 0-1 Programming. *Management Science* 26, 86–96, 1980.

[3] E. Balas, S. Schmieta, C. Wallace. Pivot and Shift. A Mixed Integer Programming Heuristic. *Technical Report Carnegie Mellon University*, Pittsburgh, PA, December 2003.

[4] R.E. Bixby. Personal communication. 2003

[5] J.W. Chinneck. The constraint consesus method for finding approximately feasible points in nonlinear programs. *Technical Report Carleton University*, Ottawa, Ontario, Canada, October 2002.

[6] E. Danna, E. Rothberg, C. Le Paper. Exploring relaxation induced neighborhoods to improve MIP solutions. *ILOG Technical Report*, 2003.

[7] Double-Click sas. Personal communication, 2001.

[8] M. Fischetti and A. Lodi. Local Branching. *Mathematical Programming* 98, 23–47, 2003.

[9] F. Glover and M. Laguna. General Purpose Heuristics For Integer Programming: Part I. *Journal of Heuristics* 2, 343–358, 1997.

[10] F. Glover and M. Laguna. General Purpose Heuristics For Integer Programming: Part II. *Journal of Heuristics* 3, 161–179, 1997.

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

[12] F.S. Hillier. Effcient Heuristic Procedures For Integer Linear Programming With An Interior. *Operations Research* 17, 600–637, 1969.

[13] T. Ibaraki, T. Ohashi and H. Mine. A Heuristic Algorithm For Mixed-Integer Programming Problems. *Mathematical Programming Study* 2, 115–136, 1974.

[14] G.W. Klau. Personal communication, 2002.

[15] A. Løkketangen. Heuristics for 0-1 Mixed-Integer Programming. In P.M. Pardalos and M.G.C. Resende (ed.s) *Handbook of Applied Optimization*, Oxford University Press, 474–477, 2002.

[16] A. Løkketangen and F. Glover. Solving Zero/One Mixed Integer Programming Problems Using Tabu Search. *European Journal of Operational Research* 106, 624-658, 1998.

[17] M. Lübbecke. Personal communication, 2002.

[18] A.J. Miller. Personal communication, 2003.

[19] MIPLIB - Mixed Integer Problem Library 2003. `http://miplib.zib.de`

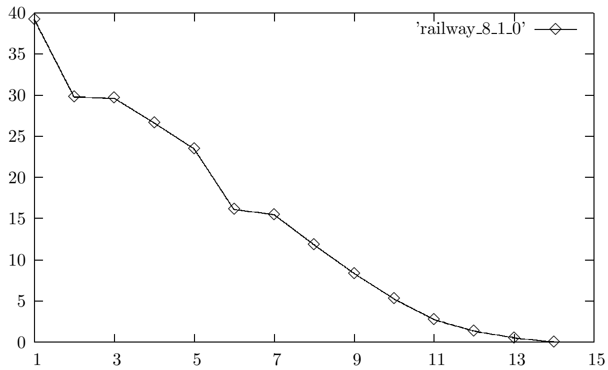
[20] M. Nediak and J. Eckstein. Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming *Research Report RRR 53-2001*, RUTCOR, Rutgers University, October 2001.

[21] J. Patel and J.W. Chinneck Active-Constraint Variable Ordering for Faster Feasibility of Mixed Integer Linear Programs. *Technical Report Carleton University*, Ottawa, Ontario, Canada, November 2003.

[22] E. Rothberg. Personal communication, 2002.

[23] E. Rothberg. Personal communication, 2003.

[24] K. Spielberg, M. Guignard. Sequential (Quasi) Hot Start Method for BB (0,1) Mixed Integer Programming. *Wharton School Research Report*, 2002.



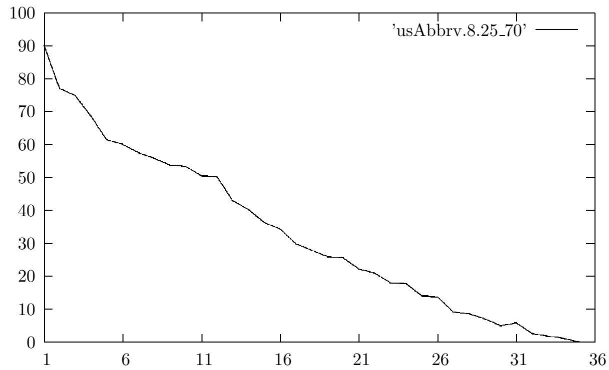


Figure 2: Plot of the infeasibility measure $\Delta(x^*, \tilde{x})$ at each pumping cycle