Using the analytic center in the feasibility pump

Daniel Baena Jordi Castro
Dept. of Stat. and Operations Research
Universitat Politècnica de Catalunya
daniel.baena@upc.edu jordi.castro@upc.edu
Research Report UPC-DEIO DR 2010-03
August 2010; updated May 2011, June 2011

Report available from http://www-eio.upc.es/~jcastro

Using the analytic center in the feasibility pump

Daniel Baena[a], Jordi Castro[*,a]

[a] *Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain*

Abstract

The feasibility pump (FP) has proved to be a successful heuristic for finding feasible solutions of mixed integer linear problems. Briefly, FP alternates between two sequences of points: one of feasible solutions for the relaxed problem, and another of integer points. This short paper extends FP, such that the integer point is obtained by rounding a point on the (feasible) segment between the computed feasible point and the analytic center for the relaxed linear problem.

Key words: Analytic Center, Interior-point Methods, Mixed-integer Linear Programming, Feasibility Problem, Primal Heuristics

1. Introduction

The problem of finding a feasible solution of a generic mixed integer linear problem (MILP) of the form

$$\begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s. to} & Ax = b \\ & x \geq 0 \\ & x_j \quad \textit{integer} \quad \forall j \in \mathcal{I}, \end{array} \tag{1}$$

where $A \in \mathbb{R}^{mxn}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$ and $\mathcal{I} \subseteq \mathcal{N} = \{1, \ldots, n\}$, is a NP-hard problem. In [5, 7] the authors proposed a new heuristic approach to compute MILP solutions, named the *feasibility pump* (FP). This heuristic turned out to be successful in finding feasible solutions even for some hard MILP instances. A slight modification of FP was suggested in [1], named the *objective feasibility pump*, in order to improve the quality of the solutions in terms of the objective value. The main difference between the two versions is that the objective FP, in contrast to the original version, takes the objective function of the MILP into account during the execution of the algorithm. FP alternates between feasible (for the linear relaxation of MILP) and integer points, hopefully converging to a feasible integer solution. The integer point is obtained by applying some rounding procedure to the feasible solution. This paper suggests an extension of FP where all the points in a feasible segment are candidates to be rounded. The end points of this segment are the feasible point of the standard or objective FP and some interior point of the polytope of the relaxed problem, the analytic center being the best candidate (our approach will be named analytic center FP, or AC-FP). When the end point of the segment in the boundary of the polytope is considered for rounding, we obtain the standard FP algorithm. The motivation of this approach is that rounding a point of the segment closer to the analytic center may increase the chances of obtaining an integer point in some instances, thus reducing the number of FP iterations. The computational results with AC-FP show that, for some instances, taking a point in the interior of the feasible segment may be more effective than the standard end point of the objective FP. A recent version of FP [8] introduced a new improved rounding scheme based on constraint propagation. Although in this work we considered as base code a freely available implementation of the objective FP, AC-FP could also be used with the new rounding scheme of [8].

[*]Corresponding address: Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Campus Nord, Office C5218, Jordi Girona 1–3, 08034 Barcelona, Catalonia, Spain.

Email addresses: daniel.baena@upc.edu (Daniel Baena), jordi.castro@upc.edu (Jordi Castro)

1. initialize $t := 0$ and $x^* := \arg\min\{c^T x : Ax = b, x \geq 0\}$
2. **if** $x^*_{\mathcal{I}}$ is integer **then** return(x^*) **end if**
3. $\tilde{x} := [x^*]$ (rounding of x^*)
4. **while** time < TimeLimit **do**
5. $\quad x^* := \arg\min\{\triangle\ (x, \tilde{x}) : Ax = b, x \geq 0\}$
6. $\quad$ **if** $x^*_{\mathcal{I}}$ is integer **then** return(x^*) **end if**
7. $\quad$ **if** $\exists j \in \mathcal{I} : [x^*_j] \neq \tilde{x}_j$ **then**
8. $\quad\quad \tilde{x} := [x^*]$
9. $\quad$ **else**
10. $\quad\quad$ restart
11. $\quad$ **end if**
12. $\quad t := t + 1$
13. **end while**
14. return(FP failed)

Figure 1: The feasibility pump heuristic (original version)

Interior-point methods have been applied in the past in branch-and-bound frameworks for MILP and mixed integer nonlinear problems (MINLP) [3, 4, 11, 12]. However, as far as we know, the only previous attempt to apply them to a primal heuristic was [13]. (We were warned about the existence of this work—when it was a manuscript under review—by an anonymous reviewer. That and our approach were independently developed, although they share the idea of using the analytic center for obtaining MILP feasible solutions.) Although AC-FP and the approach of [13] (named analytic center feasibility method (ACFM)) have the same motivation (using the analytic center for getting MILP feasible solutions), both approaches are significantly different, as shown at the end of Subsection 2.2. Briefly, (i) AC-FP relies on FP, while ACFM is based on an analytic center cutting plane method; (ii) AC-FP only computes one analytic center, while ACFM computes one per iteration; (iii) as a consequence of the previous point, ACFM can be computationally expensive, while AC-FP is almost as efficient as FP.

The paper is organized as follows. The remainder of Section 1 reviews the original FP version of [5, 7], the modified objective FP of [1], and the ACFM of [13]. Section 2 introduces AC-FP, showing it is an extension of FP, and comparing it with ACFM. Finally, Section 3 reports computational results on a subset of MILP instances from MIPLIB 2003 [2], comparing the objective FP, ACFM and AC-FP.

1.1. The original feasibility pump

The FP heuristic starts by solving the linear programming (LP) relaxation of (1)

$$\min_x\{c^T x \ : \ Ax = b, x \geq 0\}, \tag{2}$$

and its solution x^* is rounded to an integer point $\tilde{x}$, which may be infeasible for (2). The rounding $\tilde{x}$ of a given x^*, denoted as $\tilde{x} = [x^*]$, is obtained by setting $\tilde{x}_j = [x^*_j]$ if $j \in \mathcal{I}$ and $\tilde{x}_j = x^*_j$ otherwise, where $[\bullet]$ represents scalar rounding to the nearest integer. If $\tilde{x}$ is infeasible, FP finds the closest $x^* \in P$, where

$$P = \{x \in \mathbb{R}^n \ : \ Ax = b, x \geq 0\}, \tag{3}$$

by solving the following LP

$$x^* = \arg\min\{\triangle\ (x, \tilde{x}) : Ax = b, x \geq 0\}, \tag{4}$$

$\triangle\ (x, \tilde{x})$ being defined (using the L_1 norm) as

$$\triangle\ (x, \tilde{x}) = \sum_{j \in \mathcal{I}} |x_j - \tilde{x}_j|. \tag{5}$$

Notice that continuous variables $\tilde{x}_j$, $j \notin \mathcal{I}$, do not play any role. If $\triangle\ (x^*, \tilde{x}) = 0$ then $x^*_j (= \tilde{x}_j)$ is integer for all $j \in \mathcal{I}$, so x^* is a feasible solution for (1). If not, FP finds a new integer point $\tilde{x}$ from x^* by rounding. The pair of points $(\tilde{x}, x^*)$ with $\tilde{x}$ integer and $x^* \in P$ are iteratively updated at each FP iteration with the aim of reducing as much as possible the distance $\triangle\ (x^*, \tilde{x})$. An outline of the FP algorithm is showed in Figure 1. To avoid that the procedure gets stuck at the same sequence of integer and feasible, there is a restart procedure when the previous integer point $\tilde{x}$ is revisited (lines 7–11 of algorithm of Figure 1). In a restart, a random perturbation step is performed.

The FP implementation has three stages. *Stage 1* is performed just on the binary variables by relaxing the integrality conditions on the general integer variables. In *stage 2* FP takes all integer variables into account. The FP algorithm exits stage 1 and goes to stage 2 when either (a) a feasible point with respect to only the binary variables has been found; (b) the minimum $\triangle\ (x^*, \tilde{x})$ was not updated during a certain number of iterations; or (c) the maximum number of iterations was reached. The point $\tilde{x}$ that produced the smallest $\triangle\ (x^*, \tilde{x})$ is stored and passed to stage 2 as the initial $\tilde{x}$ point. When FP turns out to be unable to find a feasible solution within the provided time limit, the default procedure of the underlying MILP solver (CPLEX 12 [10] in this work) is started; this is named *stage 3*.

1.2. The modified objective feasibility pump

According to [1], although the original FP heuristic of [5, 7] has proved to be a very successful heuristic for finding feasible solutions of mixed integer programs, the quality of their solutions in terms of objective value tends to be poor. In the original FP algorithm of [5, 7] the objective function of (1) is only used at the beginning of the procedure. The purpose of the objective FP [1] is, instead of instantly discarding the objective function of (1), to consider a convex combination of it and $\triangle\ (x, \tilde{x})$, reducing gradually the influence of the objective term. The hope is that FP still converges to a feasible solution but it concentrates the search on the region of high-quality points. The modified objective function $\triangle_\alpha\ (x, \tilde{x})$ is defined as

$$\triangle_\alpha\ (x, \tilde{x}) := (1 - \alpha)\ \triangle\ (x, \tilde{x}) + \alpha \frac{\| \triangle \|}{\|c\|} c^T x, \quad \alpha \in [0, 1], \tag{6}$$

where $\| \cdot \|$ is the Euclidean norm of a vector, and $\triangle$ is the objective function vector of $\triangle\ (x, \tilde{x})$ (i.e., at stage 1 is the number of binary variables, and at stage 2 is the number of integer (both general integer and binary) variables). At each FP iteration α is geometrically decreased with a fixed factor $\varphi < 1$, i.e., $\alpha_{t+1} = \varphi\alpha_t$ and $\alpha_0 \in [0, 1]$. Notice that the original FP algorithm is obtained using $\alpha_0 = 0$. The objective FP algorithm is basically the same as the original FP algorithm of Figure 1, replacing $\triangle\ (x, \tilde{x})$ by $\triangle_{\alpha_t}\ (x^*, \tilde{x})$ at line 5, performing at the beginning the initialization of α_0, and adding at the end of the loop $\alpha_{t+1} = \varphi\alpha_t$.

1.3. The analytic center feasibility method (ACFM)

ACFM [13] is a three-phase procedure that mainly relies on the analytic center cutting plane method. In phase-I it computes (i) the analytic center $\bar{x}$ of the bounded polyhedron

$$P \cap \{x : c^T x \leq z\} \cap C, \tag{7}$$

z being an upper bound on the objective function and C a set of valid cuts (initially empty), and (ii) the minimizer $x^*_{\min}$ and maximizer $x^*_{\max}$ of the objective function $c^T x$ subject to $x \in P$. Actually, the formulation in [13] of the problem for computing the analytic center is different from the above one, since it considers only inequalities, and it needs a reformulation of equality constraints; our approach, detailed in Section 2 below, directly works with the original formulation of the problem, as it can deal with equality constrained problems. Scanning the segments $\overline{\bar{x}\, x^*_{\min}}$ and $\overline{\bar{x}\, x^*_{\max}}$, phase-I tries to obtain the closest integer point to the analytic center by rounding the integer components of different segment points—let us name $\tilde{x}$ such a rounded point—and adjusting the remaining continuous components by solving

$$\min_x \{c^T x\ :\ Ax = b, x \geq 0, x_j = \tilde{x}_j\ j \in \mathcal{I}\}. \tag{8}$$

If (8) is feasible then an integer feasible solution is obtained. Whether this problem is feasible or not, phase-II is started. If phase-I found a feasible integer point, the upper bound z on the objective is updated and we go to phase-I again, to recompute the new analytic center (different from previous iteration, since z, thus (7), changed). If no feasible integer point was found at phase-I, then additional constraints (cuts) are added to C to move the analytic center towards the interior of the integer feasible region, and phase-I is restarted again (computing a new analytic center for the new polyhedron (7)). The procedure iterates Phase-I and Phase-II until some stopping criteria is satisfied (iteration limit—20 iterations in [13]—, or quality of the solution). If no feasible solution is found the procedure switches to a phase-III which is similar to the stage 3 of FP.

2. The analytic center feasibility pump (AC-FP)

2.1. The analytic center

Given the LP relaxation (2), its analytic center is defined as the point $\bar{x} \in P$ that minimizes the *primal potential function* $-\sum_{i=1}^{n} \ln x_i$, i.e.,

$$\begin{array}{rl} \bar{x} = \arg\min\limits_{x} & -\sum_{i=1}^{n} \ln x_i \\ \text{s. to} & Ax = b \\ & x > 0. \end{array} \tag{9}$$

Note that the analytic center is well defined only if P is bounded. Note also that constraints $x > 0$ could be avoided, since the domain of ln are the positive numbers. Problem (9) is a linearly constrained strictly convex optimization problem. It is easily seen that the $\arg\min -\sum_{i=1}^{n} \ln x_i$ is equivalent to the $\arg\max \prod_{i=1}^{n} x_i$. Therefore, the analytic center provides the point that maximizes the distance to the hyperplanes $x_i = 0, i = 1, \ldots, n$, and it is thus expected to be well centered in the interior of the polytope P. We note that the analytic center is not a topological property of a polytope, and it depends on how the polytope is represented. That is, two different sets of linear inequalities P and P' defining the same polytope may provide different analytic centers. Other centers, such as the center of gravity, are not affected by different formulations of the same polyhedron (but they are computationally more expensive). In this sense, redundant inequalities may change the location of the analytic center (i.e., if formulation P' is obtained from formulation P by adding redundant constraints, it will provide a different analytic center). Additional details can be found in [14].

The analytic center may be computed by solving the KKT conditions of (9)

$$\begin{array}{rcll} Ax & = & b & \\ A^T y + s & = & 0 & \\ x_i s_i & = & 1 & i = 1, \ldots, n \\ (x, s) & > & 0, & \end{array} \tag{10}$$

$y \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ being the Lagrange multipliers of $Ax = b$ and $x > 0$ respectively. Alternatively, and in order to use an available highly efficient implementation, the analytic center was computed in this work by applying a primal-dual path-following interior-point algorithm to the barrier problem of (2), after removing the objective function term (i.e., setting $c = 0$):

$$\begin{array}{rl} \min\limits_{x} & -\mu \sum_{i=1}^{n} \ln x_i \\ \text{s. to} & Ax = b \\ & x > 0, \end{array} \tag{11}$$

where μ is a positive parameter (the parameter of the barrier) that tends to zero. The arc of solutions of (11) $x^*(\mu)$ is named the primal central path. The central path converges to the analytic center of the optimal set. When $c = 0$ (as in (11)) the central path converges to the analytic center of the feasible set P [14].

2.2. Using the analytic center in the feasibility pump heuristic

Once the analytic center has been computed, it can be used to (in theory infinitely) increase the number of feasible points candidates to be rounded. Instead of rounding, at each FP iteration, the feasible point $x^* \in P$, points on the segment

$$x(\gamma) = \gamma \bar{x} + (1 - \gamma) x^* \quad \gamma \in [0, 1] \tag{12}$$

will be considered. Note that the segment is feasible, since it is a convex combination of two feasible points.

AC-FP first considers a *stage 0* (which is later applied at each FP iteration) where several $x(\gamma)$ points are tested, from $\gamma = 0$ to $\gamma = 1$ (i.e, from x^* to $\bar{x}$). Each $x(\gamma)$ is rounded to $\tilde{x}(\gamma)$. If $\tilde{x}(\gamma)$ is feasible, then a feasible integer solution was found and the procedure is stopped at the stage 0. Otherwise the algorithm proceeds with the next stage of FP, considering two different options:

a) using the point $\tilde{x}(0) = [x^*]$ (option $\gamma = 0$);

1. initialize $t := 0$, $\alpha_0 \in [0,1]$, $\varphi \in [0,1]$, and $x^* := \arg\min\{c^T x : Ax = b, x \geq 0\}$
2. compute analytic center $\bar{x} := \arg\min\left\{-\sum_{i=1}^{n} \ln x_i : Ax = b, x > 0\right\}$
3. { *Beginning of stage 0*}
4. **for** $\gamma \in [0,1]$ **do**
5. $\quad x(\gamma) := \gamma\bar{x} + (1-\gamma)x^*$
6. $\quad \tilde{x}(\gamma) := [x(\gamma)]$ (rounding of $x(\gamma)$)
7. $\quad$ **if** $\tilde{x}(\gamma)$ is feasible **then** return($\tilde{x}(\gamma)$) **end if**
8. **end for**
9. { *End of stage 0*}
10. select $\hat{x}$ from the set $\{\tilde{x}(\gamma)\}$
11. **while** time < TimeLimit **do**
12. $\quad x^* := \arg\min\{\triangle_{\alpha_t}(x, \hat{x}) : Ax = b, x \geq 0\}$
13. $\quad$ **for** $\gamma \in [0,1]$ **do**
14. $\quad\quad x(\gamma) := \gamma\bar{x} + (1-\gamma)x^*$
15. $\quad\quad \tilde{x}(\gamma) := [x(\gamma)]$ (rounding of $x(\gamma)$)
16. $\quad\quad$ **if** $\tilde{x}(\gamma)$ is feasible **then** return($\tilde{x}(\gamma)$) **end if**
17. $\quad$ **end for**
18. $\quad$ select $\hat{x}$ from the set $\{\tilde{x}(\gamma)\}$
19. $\quad$ **if** $\hat{x}_I \neq \tilde{x}_I$ **then**
20. $\quad\quad \tilde{x} := \hat{x}$
21. $\quad$ **else**
22. $\quad\quad$ restart
23. $\quad$ **end if**
24. $\quad \alpha_{t+1} := \varphi\alpha_t$
25. $\quad t := t + 1$
26. **end while**
27. return(FP failed)

Figure 2: The AC-FP heuristic

b) using the point $\tilde{x}(\gamma)$ that minimizes $\|\tilde{x}(\gamma) - x(\gamma)\|_\infty$ (option L_∞).

If the first option is applied at each FP iteration, and no feasible $\tilde{x}(\gamma)$ for $\gamma > 0$ is found, AC-FP behaves as the standard FP algorithm. In the second option, if no feasible $\tilde{x}(\gamma)$ is found, the procedure selects the $x(\gamma)$ which is closer to $[x(\gamma)]$ according to the L_∞ norm. The aim is to select the point with more chances to become both integer and feasible, in an attempt to reduce the number of FP iterations. This second option provided better results in general and it was used in the computational results of Section 3. It is worth to note that if the rounding of several $x(\gamma)$ points is feasible, the procedure selects the one with the lowest γ, i.e., the one closest to x^* (instead of the one closest to the analytic center $\bar{x}$), since this point was computed considering the objective function (for $\alpha > 0$). An outline of the algorithm is shown in Figure 2.

From Figure 2 it is clear that AC-FP only computes one analytic center (that of P) at line 2 of the algorithm, unlike ACFM [13] which computes one analytic center (for a modified polyhedron) at each iteration. This is computationally the most significant difference between AC-FP and ACFM: since the computation of analytic centers can be expensive, AC-FP is more efficient than ACFM. It is also seen that AC-FP and ACFM are completely different approaches: the former is an extension of FP, the latter is based on computing analytic centers of modified polyhedrons obtained by adding cutting planes to P.

Both procedures, AC-FP and ACFM, consider the feasible segment between the analytic center $\bar{x}$ and a solution of the relaxed problem (x^* in AC-FP, $x^*_{\min}$ and $x^*_{\max}$ in ACFM) for rounding purposes. It is worth to note that in AC-FP the analytic center is the same for all the iterations and x^* is different at each iteration, whereas the opposite holds for ACFM: it computes a different analytic center at each iteration whereas $x^*_{\min}$ and $x^*_{\max}$ are uniquely determined at the beginning. In addition, AC-FP and ACFM use the rounded point $\tilde{x}(\gamma)$ in a different manner. AC-FP checks if $\tilde{x}(\gamma)$ is feasible, and stops the procedure once the first feasible $\tilde{x}(\gamma)$ is found (which is indeed the criterion considered by FP). On the other hand, ACFM, which may obtain a rounded feasible point at its phase-I, keeps on iterating with phase-I and phase-II until some stopping criteria (i.e., time limit or quality of the solution) is satisfied. In addition, after obtaining the rounded point, ACFM solves (8) for adjusting the remaining continuous components (this is not done by AC-FP, which relies on the overall FP procedure for performing a similar adjustment at line 12 of the algorithm of Figure 2). Since AC-FP may obtain a feasible point at stage 0 close to the analytic center $\tilde{x}$ and far from the feasible point $x^* \in P$, this point may provide a very large objective function value. An extension would be to save this point and keep on looking for new feasible points of higher quality (as done by ACFM).

As stated in Subsection 1.3, ACFM computes two linear feasible points $x^*_{\min}$ and $x^*_{\max}$, the minimizer and maximizer of $c^T x$ within P, and it considers the two segments that join the analytic center of the current ACFM iteration

with those two points. On the other hand, AC-FP only considers one segment between $\bar{x}$ and x^*. Actually, we initially also considered two segments: the current one $\overline{\bar{x}\, x^*}$, and a second one joining $\bar{x}$ with the farthest feasible point from $\bar{x}$ in the direction $\bar{x} - x^*$ (name it x_f^*). Note that this point is easily computed as $x_f^* = \bar{x} + \beta^*(\bar{x} - x^*)$, where $\beta^* = \min\{\frac{-x_i}{(\bar{x}-x^*)_i} : (\bar{x} - x^*)_i < 0, i = 1, \ldots, n\}$. The computational benefit of using x_f^* instead of $x_{\max}^*$ is that the solution of an extra LP problem is avoided. However, in practice, using the second segment $\overline{\bar{x}\, x_f^*}$ was not useful, and it was discarded in the final AC-FP implementation.

3. Computational results

AC-FP was implemented using the base code of the objective FP, freely available from `http://www.or.deis.unibo.it/research_pages/ORcodes/FP-gen.html`. The base FP implementation was extended for computing the analytic center using three different interior-point solvers, CPLEX [10], GLPK [9] and PCx [6]. The new code is available from `http://www-eio.upc.es/~dbaena/sw/2010/fp_analytic_center.tgz`. CPLEX integrates better with the rest of the FP code, which also relies on CPLEX, and it also turned out to be significantly more efficient than GLPK and PCx. On the other hand, even deactivating all the preprocessing options and removing the crossover postprocess, CPLEX was not always able to provide the analytic center of P because of its aggressive reduced preprocessing (which can not be deactivated as we were told by CPLEX developers). For instance, for $P = \{x : \sum_{i=1}^{n} x_i = n, x \geq 0\}$, the barrier option of CPLEX did not apply the interior-point algorithm, not providing an interior solution (i.e., it provided $x_i = n$, $x_j = 0$, $j \neq i$), whereas both GLPK and PCx reported the right analytic center $x_i = 1, i = 1, \ldots, n$. Of the other two solvers, PCx turned out to be much more efficient than GLPK. Indeed, PCx may handle upper bounds implicitly (i.e., $0 \leq x \leq 1$ from linear relaxations of $x \in \{0, 1\}$) in its interior-point implementation, whereas GLPK transforms the problem to the standard form (replacing $x \leq 1$ by $x + s = 1, s \geq 0$), significantly increasing the size of the Newton's system to be solved at each interior-point iteration.

The AC-FP implementation was applied to a subset of MIPLIB2003 instances, whose dimensions are shown in Table 1. Columns "rows", "cols", "nnz", "int", "bin" and "con" provide respectively the number of constraints, variables, nonzeros, general integer variables, binary variables, and continuous variables of the instances. Column "objective" shows the optimal objective function. Unknown optimal objectives are marked with a "?".

Table 2 shows the results obtained with AC-FP using PCx and CPLEX-12.1. For the two AC-FP variants, Table 2 reports the number of FP iterations (columns "niter"), the objective value of the feasible point found ("fobj"), the gap between the feasible and the optimal solution ("gap%"), and the FP stage where the feasible point was found ("stage"). Columns "tFP(tAC)" report separately the CPU time spent in stages 1 to 3 ("tFP") and the time for computing the analytic center before stage 0 (in brackets, "tAC"); the total time is the sum of "tFP" and "tAC". Columns "AC value" show the value of the original objective function evaluated at the analytic center. Differences are due to different computed analytic centers because both solvers apply very distinct preprocessing strategies.

Table 3 compares AC-FP with ACFM using the subset of nine MIPLIB2003 instances solved in [13]. For ACFM, Table 3 reports the number of ACFM iterations needed to reach the feasible solution ("niter"), the feasible solution (column "fobj"), and the gap between the solution found by ACFM and the optimal solution (column "gap%"). Column "tt(tAC)" reports the total CPU time of the ACFM algorithm, including the amount of CPU time in seconds spent on calculating the analytic centers (in brackets, "tAC"). The best result (i.e., execution with the lowest gap) is highlighted in boldface.

Table 4 compares AC-FP with the objective FP. For the objective FP we report the number of FP iterations (column "niter"), the objective value of feasible point found ("fobj"), the gap between the feasible and the optimal solution ("gap%"), the FP stage where the feasible point was found ("stage") and the total CPU time (column "tt"). The best result (i.e. that with the lowest gap if obtained in stages 0–2), is highlighted in boldface. Note that for instance "swath" objective FP is considered the best approach, though the gap is greater than for AC-FP, since the solution with objective FP was found at stage 2, while AC-FP failed and it needed stage 3. This same argument was applied for instance "dano3mip", of unknown gap. For instance "liu" AC-FP with PCx provided a better objective function, though the gap is also unknown. If two approaches provide the same gap, but one is significantly more efficient, this is marked as the best result (as in instance "ds").

The default FP settings were used as suggested in [1]. All runs were carried on a Dell PowerEdge 6950 server with four dual core AMD Opteron 8222 3.0 GHZ processors (without exploitation of parallelism capabilities) and 64 GB

Instance	rows	cols	nnz	int	bin	con	objective
10teams	230	2025	12150	0	1800	225	924
a1c1s1	3312	3648	10178	0	192	3456	11503.40
aflow30a	479	842	2091	0	421	421	1158
aflow40b	1442	2728	6783	0	1364	1364	1168
air04	823	8904	72965	0	8904	0	56137
air05	426	7195	52121	0	7195	0	26374
arki001	1048	1388	20439	96	415	877	7580810
atlanta-ip	21732	48738	257532	106	46667	1965	90.00
cap6000	2176	6000	48243	0	6000	0	-2451380
dano3mip	3202	13873	79655	0	552	13321	?
danoint	664	521	3232	0	56	465	65.66
disctom	399	10000	30000	0	10000	0	-5000
ds	656	67732	1024059	0	67732	0	93.52
fast0507	507	63009	409349	0	63009	0	174
fiber	363	1298	2944	0	1254	44	405935
fixnet6	478	878	1756	0	378	500	3983
gesa2-o	1248	1224	3672	336	384	504	25779900
gesa2	1392	1224	5064	168	240	816	25779900
glass4	396	322	1815	0	302	20	1200010000
harp2	112	2993	5840	0	2993	0	-73899800
liu	2178	1156	10626	0	1089	67	?
manna81	6480	3321	12960	3303	18	0	-13164
markshare1	6	62	312	0	50	12	1
markshare2	7	74	434	0	60	14	1
mas74	13	151	1706	0	150	1	11801.20
mas76	12	151	1640	0	150	1	40005.10
misc07	212	260	8619	0	259	1	2810
mkc	3411	5325	17038	0	5323	2	-563.84
mod011	4480	10958	22254	0	96	10862	-54558500
modglob	291	422	968	0	98	324	20740500
msc98-ip	15850	21143	92918	53	20237	853	19839500
mzzv11	9499	10240	134603	251	9989	0	-21718
mzzv42z	10460	11717	151261	235	11482	0	-20540
net12	14021	14115	80384	0	1603	12512	214
noswot	182	128	735	25	75	28	-41
nsrand-ipx	735	6621	223261	0	6620	1	51200
nw04	36	87482	636666	0	87482	0	16862
opt1217	64	769	1542	0	768	1	-16
p2756	755	2756	8937	0	2756	0	3124
pk1	45	86	915	0	55	31	11
pp08aCUTS	246	240	839	0	64	176	7350
pp08a	136	240	480	0	64	176	7350
protfold	2112	1835	23491	0	1835	0	-31
qiu	1192	840	3432	0	48	792	-132.87
roll3000	2295	1166	29386	492	246	428	12890
rout	291	556	2431	15	300	241	1077.56
set1ch	492	712	1412	0	240	472	54537.80
seymour	4944	1372	33549	0	1372	0	423
sp97ar	1761	14101	290968	0	14101	0	660706000
swath	884	6805	34965	0	6724	81	467.40
timtab1	171	397	829	94	64	239	764772
timtab2	294	675	1482	164	113	398	1096560
tr12-30	750	1080	2508	0	360	720	130596
vpm2	234	378	917	0	168	210	13.75

?: Unknown value

Table 1: Characteristics of the subset of MILP instances from MIPLIB 2003

	AC-FP with PCx						AC-FP with CPLEX					
Instance	niter	fobj	tFP(tAC)	stage	gap%	AC value	niter	fobj	tFP(tAC)	stage	gap%	AC value
10teams	179	1022	26(0)	3	10.59	1020	177	1056	25(0)	3	14.27	1020
a1c1s1	0	46756.40	0(0)	0	306.43	50396.80	0	38193.60	0(0)	0	232	40504.70
aflow30a	171	3802	1(0)	2	228.13	5377.34	298	5578	2(0)	2	381.36	4714.70
aflow40b	394	8300	12(0)	3	610.09	7234.03	54	7051	2(0)	1	503.25	6635.6
air04	186	72098.00	1220(2)	3	28.43	79260.30	186	71223.99	1147(0)	3	26.87	79989.5
air05	186	37907	162(1)	3	43.73	45309	190	35798	148(0)	3	35.73	45732.10
arki001	871	7729296.21	43(0)	3	1.96	7807100	1573	7763720.15	79(0)	3	2.41	7822170
atlanta-ip	42	198.02	68(9398)	1	118.68	171.31	397	154.01	934(11)	3	70.32	159.76
cap6000	0	-2442800	1(0)	0	0.35	-596562	0	-2442800	1(0)	0	0.35	-109362
dano3mip	205	1000	1892(17)	3	?	12849.20	252	1000	1947(4)	3	?	995.15
danoint	99	76	4(0)	1	15.50	434.456	230	85.50	9(0)	3	29.75	66.77
disctom	4	-5000	3(1)	1	0	-5000	4	-5000	4(0)	1	0	-5000
ds	198	5418.56	1945(10)	3	5633.77	1053.93	0	5418.56	1(2)	0	5633.77	5418.56
fast0507	39	11884	131(4)	1	6691.43	8254.52	0	275	2(1)	0	57.71	122425
fiber	41	6481510	0(0)	1	1496.68	19694200	15	3147830	0(0)	1	675.45	45602200
fixnet6	18	38401	0(0)	1	863.91	60883.20	0	97271.70	0(0)	0	2341.58	101827
gesa2-o	20	71213100	0(0)	2	176.23	116914000	35	32635500	1(0)	2	26.59	166784000
gesa2	3	38472300	1(0)	2	49.23	124095000	47	40307000	1(0)	2	56.35	188208000
glass4	254	10500117800	2(0)	3	775	142889000000	224	5000046800	1(0)	3	316.67	8862840000
harp2	178	-40631391	3(0)	3	45.02	-50758200	59	-49759800	1(0)	1	32.67	-46262500
liu	119	3036	4(5)	1	?	9218.57	121	5876	5(0)	1	?	959.02
manna81	0	-12948	0(6)	0	1.64	-7307.16	0	-12878	0(0)	0	2.17	0
markshare1	65	603	0(0)	1	30100	30.48	0	7286	0(0)	0	364250	7286
markshare2	66	925	0(0)	1	46200	36.10	0	10512	0(0)	0	525550	10512
mas74	0	57195600000	0(0)	0	484618022.50	571956000000	0	50000000000	0(0)	0	423649728.01	1000000000000
mas76	0	26804400000	0(0)	0	67000682.38	536000000000	0	50000000000	0(0)	0	124980840.41	1000000000000
misc07	217	3935	3(0)	2	40.02	3601.66	219	3410	2(0)	2	21.34	4894.40
mkc	13	-276.96	1(0)	1	50.79	-253.58	12	38.81	1(1)	1	106.69	-95.53
mod011	23	-37482400	3(1)	1	31.30	-31430100	23	-35547800	3(0)	1	34.84	-36600000
modglob	60	21809700	1(0)	1	5.16	272286000	0	82243300	0(0)	0	296.53	142949000
msc98-ip	33	30196300	16(949)	1	52.20	29571000	29	30928000	19(22)	1	55.89	29545100
mzzv11	567	-16262	435(116)	3	25.12	-4264.40	561	-13744	484(7)	3	36.71	-4794.93
mzzv42z	23	-12736	13(147)	1	37.99	-3210.77	27	-14192	12(15)	1	30.90	-3825.70
net12	25	337	10(86)	1	57.21	325.12	25	337	8(27)	1	57.21	337
noswot	34	-15	0(0)	2	61.90	-21.82	33	-31	0(0)	2	23.81	-15.67
nsrand-ipx	883	258080	367(2)	3	404.05	761986	694	203040	265(0)	3	296.56	802647
nw04	2	18380	9(8)	1	9	50318.90	42	61640	120(2)	1	265.54	52460.90
opt1217	124	-12.11	0(0)	1	22.80	-8.23	0	0	0(0)	0	94.12	0
p2756	244	51338	7(0)	3	1542.85	139225	279	51338	7(0)	3	1542.85	164724
pk1	57	86	0(0)	1	625	34.13	0	731	0(0)	0	6000	731
pp08aCUTS	11	16390	0(0)	1	122.98	18715	0	21671.40	0(0)	0	194.82	23012.40
pp08a	15	15850	0(0)	1	115.63	21666.70	0	18439.30	0(0)	0	150.85	18778.70
protfold							307	-18.90	365(2)	3	37.81	-18.42
qiu	41	868.57	1(0)	1	748.05	722.04	0	3693.35	0(0)	0	2858.10	4188.61
roll3000	818	40048.40	65(1)	3	210.68	44336.80	175	18507	11(1)	2	43.57	38004.10
rout	79	1644.41	1(0)	1	52.56	1455.68	74	1337.27	1(0)	1	24.08	1474.90
set1ch	0	268719	0(0)	0	392.71	224714	0	216475	0(0)	0	296.92	262834
seymour	39	754	5(35)	1	78.07	728.54	0	588	0(0)	0	38.92	1345
sp97ar	63	1161990000	57(4)	1	75.87	8272000000	97	11702100000	88(1)	1	1671.15	18441700000
swath	795	34774.58	96(1)	3	7324.22	1470.14	795	34774.58	100(0)	3	7324.22	1470.14
timtab1	169	1081000	1(0)	2	41.35	1475570	819	1401240.99	3(0)	3	83.22	419539
timtab2	972	2105005.99	6(0)	3	91.96	2052380	1072	1772242.99	7(0)	3	61.62	671850
tr12-30	214	289227.99	7(0)	3	121.47	135860	221	285716	6(0)	3	118.78	75936.50
vpm2	11	29.50	0(0)	1	106.78	48.48	27	23.75	0(0)	1	67.8	14.08

?: Unknown value

Table 2: Computational results using AC-FP with PCx and CPLEX

Instance	AC-FP with PCx		AC-FP with CPLEX		ACFM			
	tFP(tAC)	gap%	tFP(tAC)	gap%	niter	fobj	tt(tAC)	gap%
mas74	0(0)	484618022.50	0(0)	423649728.01	7	15026.47	8.89(8.26)	**434.75**
mas76	0(0)	67000682.38	0(0)	124980840.41	1	44877.42	2.55(2.1)	**12.18**
misc07	3(0)	40.02	2(0)	**21.34**	13	4795	9.28(8.71)	70.64
noswot	0(0)	61.90	0(0)	23.81	3	-37	2.51(2.11)	**9.76**
pk1	0(0)	625	0(0)	6000	1	28.99	0.75(0.72)	**163.55**
pp08aCUTS	0(0)	122.98	0(0)	194.82	1	8458	2.81(2.25)	**15.07**
pp08a	0(0)	115.63	0(0)	150.85	1	9048.56	2.07(1.5)	**23.11**
rout	1(0)	52.56	1(0)	24.08	4	1111.88	101.95(100.58)	**3.18**
vpm2	0(0)	106.78	0(0)	67.8	6	15.5	28.43(27.31)	**12.73**

Table 3: Comparison of AC-FP (PCx and CPLEX) with ACFM only for the instances solved in [13]

Instance	AC-FP with PCx		AC-FP with CPLEX		objective FP				
	tFP(tAC)	gap%	tFP(tAC)	gap%	niter	fobj	tt	stage	gap%
Problems with only binary variables									
10teams	26(0)	10.59	25(0)	14.27	278	1014	19	3	**9.73**
a1c1s1	0(0)	306.43	0(0)	232	351	22714.68	8	2	**97.45**
aflow30a	1(0)	228.13	2(0)	381.36	41	2355	0	1	**103.28**
aflow40b	12(0)	610.09	2(0)	503.25	21	2329	1	1	**99.32**
air04	1220(2)	28.43	1147(0)	26.87	45	58229	181	1	**3.73**
air05	162(1)	43.73	148(0)	35.73	3	26930	2	1	**2.11**
cap6000	1(0)	**0.35**	1(0)	**0.35**	31	-2442163	0	1	0.38
dano3mip	1892(17)	?	1947(4)	?	70	763.97	361	1	**?**
danoint	4(0)	15.50	9(0)	29.75	96	74	3	1	**12.50**
disctom	3(1)	**0**	4(0)	**0**	3	-5000	3	1	**0**
ds	1945(10)	5633.77	1(2)	**5633.77**	446	5418.56	9495	3	5633.77
fast0507	131(4)	6691.43	2(1)	57.71	8	184	51	1	**5.71**
fiber	0(0)	1496.68	0(0)	**675.45**	41	6481506.12	0	1	1496.68
fixnet6	0(0)	**863.91**	0(0)	2341.58	67	41304	0	1	936.77
glass4	2(0)	775	1(0)	**316.67**	374	12700154400	1	3	958.34
harp2	3(0)	45.02	1(0)	32.67	138	-60669440	3	1	**17.90**
liu	4(5)	**?**	5(0)	?	119	3286	1	1	?
markshare1	0(0)	**30100**	0(0)	364250	65	725	0	1	36200
markshare2	0(0)	**46200**	0(0)	525550	65	963	0	1	48100
mas74	0(0)	484618022.50	0(0)	423649728.01	109	16534.04	0	1	**40.10**
mas76	0(0)	67000682.38	0(0)	124980840.41	106	46242.57	1	1	**15.59**
misc07	3(0)	40.02	2(0)	**21.34**	188	3690	1	1	31.31
mkc	1(0)	50.79	1(1)	106.69	13	-288.96	0	1	**48.67**
mod011	3(1)	31.30	3(0)	34.84	12	-45633967.33	1	1	**16.36**
modglob	1(0)	**5.16**	0(0)	296.53	60	22995521.33	0	1	10.87
net12	10(86)	57.21	8(27)	**57.21**	216	337	12	2	57.21
nsrand-ipx	367(2)	404.05	265(0)	**296.56**	132	211040	5	2	312.38
nw04	9(8)	9	120(2)	265.54	10	17858	10	1	**5.91**
opt1217	0(0)	22.80	0(0)	94.12	40	-16	0	1	**0**
p2756	7(0)	**1542.85**	7(0)	**1542.85**	377	51338	2	3	**1542.85**
pk1	0(0)	625	0(0)	6000	56	36	0	1	**208.33**
pp08aCUTS	0(0)	122.98	0(0)	194.82	10	8360	0	1	**13.74**
pp08a	0(0)	115.63	0(0)	150.85	11	12010	0	1	**63.39**
protfold			365(2)	**37.81**	286	-16	90	2	46.88
qiu	1(0)	748.05	0(0)	2858.10	9	160.76	0	1	**219.34**
set1ch	0(0)	392.71	0(0)	296.92	46	95845.5	0	1	**75.74**
seymour	5(35)	78.07	0(0)	38.92	7	471	3	1	**11.32**
sp97ar	57(4)	75.87	88(1)	1671.15	9	919778417.68	4	1	**39.21**
swath	96(1)	7324.22	100(0)	7324.22	395	35951.85	14	2	**7575.56**
tr12-30	7(0)	121.47	6(0)	118.78	25	164128	1	1	**25.68**
vpm2	0(0)	106.78	0(0)	67.8	12	18.25	0	1	**30.51**
Problems with binary and general integer variables									
arki001	43(0)	1.96	79(0)	2.41	803	7719381.38	15	3	**1.83**
atlanta-ip	68(9398)	118.68	934(11)	**70.32**	454	156.01	227	3	75.52
gesa2-o	0(0)	176.23	1(0)	**26.59**	33	36205441.29	1	2	40.44
gesa2	1(0)	49.23	1(0)	56.35	33	28181419.78	0	2	**9.32**
manna81	0(6)	**1.64**	0(0)	2.17	52	-12940	2	2	1.70
msc98-ip	16(949)	**52.20**	19(22)	55.89	61	30502274.00	26	1	53.75
mzzv11	435(116)	25.12	484(7)	36.71	540	-17898	127	3	**17.59**
mzzv42z	13(147)	37.99	12(15)	30.90	25	-14502	49	1	**29.39**
noswot	0(0)	61.90	0(0)	23.81	13	-41	1	2	**0**
roll3000	65(1)	210.68	11(1)	**43.57**	793	36109.80	17	3	180.12
rout	1(0)	52.56	1(0)	**24.08**	117	1652.55	0	1	53.31
timtab1	1(0)	**41.35**	3(0)	83.22	216	1400493.99	1	2	83.13
timtab2	6(0)	91.96	7(0)	**61.62**	1222	1982037.99	2	2	80.75

?: Unknown value

Table 4: Comparison with objective FP

of RAM. According to the Standard Performance Evaluation Corporation (`http://www.spec.org/`) the ratio of the performance of our machine (considering specfp2000 and specint2000) and that used in [13] is about 1.5. Therefore the CPU times in Table 3 for ACFM are those of [13] divided by 1.5.

As stated in Subsection 2.2, as a consequence of computing one analytic center per iteration, ACFM can be computationally expensive, and this is the most important difference from a practical point of view between ACFM and AC-FP. Indeed, as it can be observed in Table 3, ACFM was only tested in [13] on nine of the smaller MIPLIB instances, while we applied AC-FP to 54 (some of them much larger) instances. For example, for instance "rout" ACFM needed 101 seconds and got a solution of 1111.88 (gap of 3.18%), while AC-FP needed one second for an objective of 1337.27 (gap of 24.08%); but in other cases AC-FP outperformed ACFM both in time and objective, as in instance "misc07" where ACFM required nine seconds for an objective of 4795 (gap 70.64%), while AC-FP took two seconds for an objective of 3410 (gap 21.34%).

Although from Table 4, in general it can be concluded that AC-FP is inferior to the objective FP, there are some notable exceptions. For instance, for the 13 instances with both binary and general integer variables, AC-FP (either with PCx or CPLEX) obtained a solution with a lower gap than the objective FP in eight of the 13 instances; in some cases more efficiently and even being able to find a solution when the objective FP failed (i.e., it required stage 3), as for instances "roll3000" and "atlanta-ip" (in this latter case, however, at the expense of a very large CPU time). On the other hand, for problems with only binary variables AC-FP obtained solutions with a lower gap in very few instances. A possible explanation of this different behaviour in problems with and without general integer variables is that, for a binary problem, the only feasible integer points "close" to the segment $x(\gamma)$ are $\{0,1\}^n$, which in addition may be far from the center. For problems with general integer variables, the number of feasible integer solutions close to the analytic center will be, in general, much larger. For some problems with only binary variables, AC-FP behaved very poorly, as for "mas74" and "mas76" (it stopped at stage 0 in those cases). However, in other instances it was much more efficient obtaining the same gap that the objective FP, as for "ds". Note that for "ds" AC-FP with CPLEX obtained the feasible solution in one second at stage 0 (the other two variants failed, requiring stage 3). However, in that case CPLEX did not really compute the analytic center: it solved $\min_x\{0 : x \in P\}$ heuristically, instead of applying the barrier algorithm, as required. It thus considered a segment between two feasible solutions, none of them being the analytic center of P. Therefore, the idea of using a segment of feasible points is not restricted to the case where one of the endpoints is the analytic center, and it can be extended to more general situations.

4. Conclusions

The three approaches (FP, ACFM and AC-FP) have their own benefits and disadvantages. FP is likely the fastest approach, and in general it provides good (if not the best) solutions in most instances; however it does not exploit the concept of analytic center, which may be beneficial in some instances. ACFM seems to provide better points, but it is computationally expensive and it was only tested on small instances. AC-FP is not computationally as expensive as ACFM (it only needs to compute one analytic center), and in some MILP instances outperforms FP (either in time or quality of the solution); however, for binary problems AC-FP seems not to be competitive against FP (the analytic center seems not to be helpful when we optimize within the unit cube).

Acknowledgments

We thank an anonymous reviewer for letting us know about the approach of [13]. We also thank the authors of [13] for sending a copy of their manuscript when it was still under review. This work has been supported by grants MTM2009-08747 of the Spanish Ministry of Science and Innovation, and SGR-2009-1122 of the Government of Catalonia.

References

[1] T. Achterberg, T. Berthold, Improving the feasibility pump, Discrete Optimization 4 (2007), 77–86.
[2] T. Achterberg, G. Gamrath, T. Koch, A. Martin, The mixed integer programming library: MIPLIB 2003. http://miplib.zib.de .
[3] H.Y. Benson, Mixed integer nonlinear programming using interior-point methods, Optimization Methods and Software (2010), doi: 10.1080/10556781003799303.

[4] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuejols, I.E. Grossman, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, A.Wachter, An algorithmic framework for convex mixed integer nonlinear programs, Discrete Optimization 5 (2008), 186–204.
[5] L. Bertacco, M. Fischetti, A. Lodi, A feasibility pump heuristic for general mixed-integer problems, Discrete Optimization 4 (2007), 63–76.
[6] J.Czyzyk, S. Mehrotra, M. Wagner, S.J. Wright, PCx: an interior-point code for linear programming, Optimization Methods and Software 11 (1999) 397–430.
[7] M. Fischetti, F. Glover, A. Lodi, The Feasibility Pump, Mathematical Programming 104 (2005), 91–104.
[8] M.Fischetti, D. Salvagnin, Feasibility pump 2.0, Mathematical Programming Computation 1 (2009), 201–222.
[9] GNU, GNU Linear Programming Kit v. 4.43, 2010.
[10] IBM ILOG CPLEX 12.1, User's Manual, 2010.
[11] J.E. Mitchell, Fixing variables and generating classical cutting planes when using an interior point branch and cut method to solve integer programming problems, European Journal of Operational Research 97 (1997), 139–148.
[12] J.E. Mitchell, M.J. Todd, Solving combinatorial optimization problems using Karmarkar's algorithm, Mathematical Programming 56 (1992), 245–284.
[13] J. Naoum-Sawaya, S. Elhedhli, An interior point cutting plane heuristic for mixed integer programming, Computers & Operations Research, doi: 10.1016/j.cor.2010.12.008, in press.
[14] Y. Ye, Interior Point Algorithms. Theory and Analysis, Wiley, 1997.