Boosting the Feasibility Pump

Natashia L. Boland[1], Andrew C. Eberhard[2], Faramroze G. Engineer[1], Matteo Fischetti[3], Martin W.P. Savelsbergh[1], and Angelos Tsoukalas[2]

[1]University of Newcastle, Australia
[2]Royal Melbourne Institute of Technology, Australia
[3]University of Padova, Italy

Abstract

The Feasibility Pump (FP) has proved to be an effective method for finding feasible solutions to mixed integer programming problems. FP iterates between a rounding procedure and a projection procedure, which together provide a sequence of points alternating between LP relaxation feasible but fractional solutions, and integer but LP relaxation infeasible solutions. The process attempts to minimise the distance between consecutive iterates, producing an integer feasible solution when closing the distance between them. We investigate the benefits of enhancing the rounding procedure with a clever integer line search that efficiently explores a large set of integer points. An extensive computational study on benchmark instances demonstrates the efficacy of the proposed approach.

1 Introduction

Finding feasible solutions to a Mixed Integer Program (MIP) can be extremely challenging, yet significant strides have been made over the last few decades in doing so; see Bixby (2002) and Bixby and Rothberg (2007) for the impact of these advances on the tractability of MIP problems. This is due, in a large part, to the development of sophisticated heuristics for MIPs. These heuristics can broadly be classified as pivot based methods such as Balas and Martin (1980), Balas et al. (2004), and Eckstein and Nediak (2007), local search and meta-heuristic based techniques such as Lokketangen and

Glover (1998), Fischetti and Lodi (2003), and Danna et al. (2005), and interior point based heuristics such as Hillier (1969) and Faaland and Hillier (1979). We refer the reader to Glover and Laguna (1997a,b) and Fischetti and Salvagnin (2009) for more comprehensive surveys.

Despite the advances in heuristics for MIPs, the demand for techniques for obtaining high-quality solutions to even harder, even larger instances, and for obtaining these solutions faster persists. Recently, Fischetti et al. (2005) introduced a projection-based heuristic for MIP called the Feasibility Pump (FP). FP has proven to be quite successful in finding feasible solutions and has become a standard component of state-of-the-art (commercial) solvers. The basic FP procedure works on a pair of points x^* and $\tilde{x}$, with x^* feasible for the LP relaxation of a MIP but not necessarily integer, and $\tilde{x}$ integer but not necessarily feasible. FP iteratively updates x^* and $\tilde{x}$ with the aim of reducing the "distance" between them as much as possible to ultimately produce an integer feasible solution. Given $\tilde{x}$, the process of finding an LP feasible solution closest to $\tilde{x}$ is known as the *projection problem*. Finding the closest integer point to x^*, is known as the *transformation problem*. If the L_1 norm is used as a measure of distance (as is the case for most implementations of FP), then the projection problem can be solved as an LP that is comparable in size to the LP relaxation of the MIP, and the transformation problem can be solved simply by rounding x^*.

The success of FP has sparked a great deal of interest within the integer programming community. Almost all efforts to improve FP, including Bertacco et al. (2007), Achterberg and Berthold (2007), Hanafi et al. (2010), and De Santis et al. (2010), however, have focused on improving the projection problem, i.e., finding LP feasible points x^* that are better than those found by the original FP in terms of the objective value and/or the number of integer infeasibilities. The exception being Fischetti and Salvagnin (2009) where propagation techniques are introduced within rounding. Overall, the effort expanded to find an LP feasible solution that is close to $\tilde{x}$ versus the effort to find an integer solution that is close to x^* is lopsided though. Almost all of the computational effort is spent in the projection problem in the hope that a judiciously chosen point x^* will ultimately lead to a good integer solution through rounding. While this is true to some extent, it seems imprudent to rely solely on this process, especially when the projection procedures often result in a sequence of points that cycle, i.e., when the rounded value of x^* is also the closest integer point to the LP feasible region, and anti-cycling mechanisms typically undo the work performed by the projection procedure by moving to a point x^* that is worse off in terms of objective and/or has more integer infeasibilities. Since great lengths have been taken to achieve an LP feasible point that is close to integer, it seems only appropriate to expand more effort in searching for an integer feasible solution around this point. Therefore, in this paper, we explore the benefits of replacing rounding with a procedure that examines all rounded solutions along a line segment passing through x^*.

The idea of exploring rounded points along a line segment in search of integer feasible solutions to MIP, is not new. Hillier (1969) propose a technique which examined rounded solutions along a line segment directed towards the interior of the LP feasible region for a particular class of full-dimensional MIPs where a point in the interior could be found relatively easily using parametric analysis. The search procedure is embedded within a branch-and-bound tree where the solution to the LP relaxation of a particular node in the tree forms the starting point of the line segment, and a point in the interior of the cone formed by the set of binding constraints is chosen to be the end of the line segment. Later implementations of this procedure, including that of Jeroslow and Smith (1975) and Faaland and Hillier (1979) rely on the branch-and-bound procedure to reduce the number of integer infeasibilities. More recently, Baena and Castro (2010) and Naoum-Sawaya and Elhedhli (2011) propose the use of a similar search technique using analytic centers to determine the line segment. While Baena and Castro use these ideas within the FP, Naoum-Sawaya and Elhedhli do not, but instead use a cutting plane approach where the analytic center is computed repeatedly, each time adding a cut that separates the rounded point resulting from rounding the analytic center. Both approaches use discretization to pick points along the line segment to round.

Our primary contribution is the design and implementation of a highly efficient and highly effective enhancement of the transformation step of FP based on examining rounded solutions along a line segment. Its success is based on four main ideas: (1) efficiently exploring *all* possible rounded solutions along a line segment, (2) using effective, but easy to compute end points of the line segment, (3) extending the line segment beyond the end points and projecting it back onto the hypercube defined by the variable bounds when extending it past these bounds, and (4) applying constraint propagation at carefully chosen times during the execution. A computational study covering a large and varied set of instances, more than 1,000, shows that FP with this enhancement is able to produce a better solution than the original FP for 65% of the instances, and, maybe even more important, is able to produce a feasible solution for 12% of the instances for which the original FP failed to do so.

In what follows, we give a brief description of the original FP procedure in §2, we describe the integer line search procedure, the choice of start and end points, extending and projecting the line search to explore more integer points, and the integration of constraint propagation techniques in §3, we report the results of the comprehensive computational study in §4, and we present some final remarks and opportunities for future research in §5.

2 The Feasibility Pump

Consider the mixed integer program

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax \le b \\ l \le & x \le u \\ & x_j \text{ integer } \forall j \in I, \end{aligned}$$

where A is an $m \times n$ matrix, l and u are vectors of size n corresponding to the variable lower and upper bounds respectively, and $I \subseteq \{1, 2, \ldots, n\}$ is the index set of the variables required to be integer. Let $P = \{x : Ax \le b,\ l \le x \le u\}$ be the polyhedron defined by the constraints and the variable bounds. The basic FP is outlined in Algorithm 1.

Input : MIP: $\min\{c^T x : x \in P, x_j \text{ integer } \forall j \in I\}$
Initialize: $x^* \leftarrow \arg\min\{c^T x : x \in P\}$

```
0.1 while not termination condition do
0.2     if x* is integer feasible then return x*
0.3     x̃ ← ⌈x*⌋
0.4     if cycle detected then Perturb(x̃)
0.5     x* ← LinearProj_P(x̃)
0.6 end
```

Algorithm 1: The basic FP procedure

Here, starting with x^* corresponding to an optimal solution to the LP relaxation, at each iteration, $\tilde{x}$ is obtained by rounding x^* (denoted by $\lceil x^* \rfloor$), and a new LP feasible point is obtained through the procedure $\texttt{LinearProj}_P(x^*)$ that finds a closest (with respect to the L_1 norm) LP feasible point to $\tilde{x}$ by solving

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

When all integer variables are binary, $\Delta(x, \tilde{x})$ can easily be linearized and $\texttt{LinearProj}_P(x^*)$ can be solved as an LP over P. In the case of general integer variables, the linearization requires the introduction of additional variables and constraints. Finally, in the case that a cycle is detected, perturbation or restart techniques are invoked to recover from cycling. We refer to Bertacco et al. (2007) for further details.

More sophisticated projection schemes are proposed in the literature to find LP feasible points that are less fractional and/or that have less degradation in objective function value. For example, Achterberg and Berthold (2007) propose including objective considerations within the projection process resulting in what they call the Objective Feasibility Pump (OFP), and De Santis et al. (2010) solve projection problems with a nonlinear concave penalty term to encourage LP feasible points x^* that have fewer variables that are integer infeasible.

As mentioned in the introduction, our focus is on the transformation process. If an infeasible integer solution x^* is obtained after rounding, it may be worth exploring opportunities to fix the infeasibility rather than relying only on the projection process to find a new point that when rounded leads to an integer feasible solution. The latter approach has an important drawback: there can potentially be many integer feasible solutions close to x^* that remain unexplored. This is especially true if cycling occurs and a random restart is forced (which means much of the effort expanded to find LP feasible solutions that are reasonably close to integer is discarded). Our approach, to be described next, aims to remedy these issues.

3 Integer Line Search for the Feasibility Pump

At the heart of our approach is a procedure that efficiently explores *all* rounded solutions along a line segment. This integer line search procedure is described next. However, the success of our approach also depends on the choice of line segment, extending and projecting the line segment back onto the hypercube defined by the variable bounds, and incorporating constraint propagation in the search. These aspects will be discussed separately.

3.1 The Integer Line Search Procedure

Since there are only a finite number of points along a line segment where the rounded values of integer variables are different from any other rounded point along the line segment, finding all rounded points along the line segment can be done efficiently. For ease of exposition, we start by assuming $I = \{1, \ldots, n\}$, i.e., the pure integer case. Let x^s and x^t be the start and end point of the line segment. For each variable $i \in I$, we define

$$\Lambda_i(x^s, x^t) = \left\{0 < \lambda \leq 1 : \exists \text{integer } k \text{ s.t. } (1-\lambda)x_i^s + \lambda x_i^t = k + 0.5\right\}$$

to be the set of all convex combinations of x^s and x^t where the rounded value of variable i changes. Indeed, for $\varepsilon > 0$ small enough, we have $\lceil (1-\lambda)x_i^s + \lambda x_i^t + \varepsilon \rfloor = \lceil (1-\lambda)x_i^s + \lambda x_i^t - \varepsilon \rfloor + 1$ for all $\lambda \in \Lambda_i(x^s, x^t)$. We call these points the breakpoints along the line segment for variable i. If x^s and x^t are bounded between l and u, then there are at most a pseudo-polynomial number of such breakpoints. We next provide an "efficient" characterization of these breakpoints by first observing that $\Lambda_i(x^s, x^t)$ can be equivalently stated as

$$\Lambda_i(x^s, x^t) = \left\{0 < \lambda \leq 1 : \exists \text{integer } k \text{ s.t. } \lambda = \frac{k + 0.5 - x_i^s}{x_i^t - x_i^s}\right\},$$

and then making the following observation. If $x_i^s < x_i^t$, then the point closest to x^s along the line where the rounded value of variable i is different to that of x_i^s, corresponds to the breakpoint resulting from choosing $k = \lceil x_i^s \rfloor$. Indeed, at this breakpoint, the value of variable i is $\lceil x_i^s \rfloor + 0.5$. On the other hand, if $x_i^s > x_i^t$, then the point closest to x^s along the line where the rounded value of variable i is different to that of x_i^s, corresponds to the breakpoint resulting from choosing $k = \lceil x_i^s \rfloor - 1$. Indeed, at this breakpoint, the value of variable i is $\lceil x_i^s \rfloor - 0.5$. Thus, starting with this initial breakpoint denoted by $\bar{\lambda}_i$, where

$$\bar{\lambda}_i = \begin{cases} \frac{\lceil x_i^s \rfloor + 0.5 - x_i^s}{x_i^t - x_i^s}, & \text{if } x_i^s < x_i^t \text{ and} \\ \frac{\lceil x_i^s \rfloor - 0.5 - x_i^s}{x_i^t - x_i^s}, & \text{if } x_i^s > x_i^t, \end{cases}$$

the set of all remaining breakpoints can be characterized as follows:

$$\Lambda_i(x^s, x^t) = \left\{ 0 < \lambda \leq 1 : \lambda = \bar{\lambda}_i + \frac{k}{x_i^t - x_i^s}, \text{ and } k \text{ integer} \right\}.$$

In other words, given $\bar{\lambda}_i$, the remaining breakpoints for variable i can then be obtained by incrementing (decrementing) the value of this initial breakpoint by and integer multiple of $1/(x_i^t - x_i^s)$.

By examining all breakpoints for all variables $i \in I$, we can explore all integer points that can possibly be obtained by rounding a point along the line connecting x^s and x^t. We next show how this process can be done efficiently, without having to round individual points associated with each breakpoint.

Given $\Lambda_i(x^s, x^t)$ for all $i \in I$, we define

$$\Psi(x^s, x^t) = \left\{ \begin{array}{ll} (i_k, \lambda_k, d_k)_{k=1,\dots,K} : & \text{(i) } \lambda_k \in \Lambda_{i_k}(x^s, x^t), \\ & \text{(ii) } d_k = \begin{cases} 1, & \text{if } x_{i_k}^s < x_{i_k}^t, \\ -1, & \text{otherwise, and} \end{cases} \\ & \text{(iii) } \lambda_k \leq \lambda_{k+1} \end{array} \right\}$$

to be the set of all 3-tuples consisting of variable index, breakpoint, and the indication of the change in variable along the line (i.e., increasing or decreasing), ordered by their distance from x^s. The line search procedure that explores all rounded solutions along the line segment connecting x^s and x^t is outlined in Algorithm 2.

Input : x^s and x^t
Initialize: $x \leftarrow \lceil x^s \rfloor$;
compute breakpoints $\Psi(x^s, x^t) = \{(i_k, \lambda_k, d_k)_{k=1,\dots,K}\}$;

forall $k = 1, \dots, K$ **do**
  $x_{i_k} \leftarrow x_{i_k} + d_k$
  if x *is an* ***incumbent*** **then**
    record x;
  end
end

Algorithm 2: The integer line search procedure.

The above procedure essentially creates an ordering of some subset of the integer variables, and a direction in which each of these variables changes, i.e., an indication of whether the variable increases or decreases along the line. Starting with the initial rounded solution $\lceil x^s \rfloor$, the line search procedure changes the values of individual variables by a unit amount in the appropriate direction and in the given sequence, checking the feasibility of the resulting integer point after each change in value.

The efficiency of the prescribed approach stems from the fact that the difference between rounded values corresponding to consecutive breakpoints is only one variable, and only by a unit amount. Thus, all integer points along the line can be explored by changing the value of the appropriate variables by a unit amount one at a time in the given sequence. Since all breakpoints along the line that could possibly lead to a feasible solution are considered, Algorithm 2 clearly explores all integer points that can be obtained from rounding some point along the line connecting x^s and x^t. Since we have a pseudo-polynomial characterization of points in $\Lambda(x^s, x^t)_i$ for each $i \in I$, Algorithm 2 is pseudo-polynomial in the size of $|x^t - x^s|$.

Although Hillier (1969) does not provide an explicit characterization of breakpoints, he does allude to the fact that such an efficient procedure is possible. That said, the computational results reported in Hillier (1969) and Faaland and Hillier (1979), report greater success with exploring the line segment with fixed increments together with a form of neighborhood search.

Finally, note that in the presence of continuous variables, we can solve a LP to obtain the values of the continuous variables for fixed values of the integer variables. Moreover, changing the value of the integer variables simply translates to changing the right hand side of the LP that needs to be solved. Hence, solving for the continuous variables can be done efficiently within the line search procedure with warms starts. That said, in some cases, the number of breakpoints along a line can be quite large, leading to a large number of integer points and thus a large number of LPs that need to be solved, which can become expensive even with warm starts.

3.2 The Choice of Start and End Points

If the integer line search procedure is to be a substitute for rounding in FP, then we may assume that we have available to us a fractional point x^* that is a reasonable proxy for a good integer solution. This may be a fractional solution to the LP relaxation of a node in the branch-and-bound tree or, alternatively, a point obtained in FP by projecting an integer infeasible point onto the LP relaxation. With x^* as the starting point of the line search, the procedure outlined in Algorithm 2 can be used to explore a sequence of integer points that have a greater chance of being feasible. To obtain such a sequence, the end point must be appropriately chosen to provide a compromise between integrality, objective, and feasibility considerations. To this end, since x^* is typically a good qualifier for objective and integrality considerations, the end point must be chosen so that the line segment provides a direction towards feasibility.

Ideally, we would like to obtain a point in the interior of the convex hull of integer solutions to construct the line segment. Since this is just as hard as solving the original problem, we settle for an interior point of P. An obvious choice for an interior point of P is the analytic center, which is obtained by solving the following non-linear program:

$$\min \sum_{i \in Q_1} -\mathbf{ln}(b_i - a^i x) + \sum_{i \in Q_2} (-\mathbf{ln}(x_i - l_i) - \mathbf{ln}(u_i - x_i))$$

s.t.

$$a^i x \leq b_i \text{ for all } i = 1, \ldots, m$$

$$l_i \leq x_i \leq u_i \text{ for all } i = 1, \ldots, n,$$

where $Q_1 \subseteq \{1, \ldots, m\}$ is the subset of linear constraints $a^i x \leq b_i$ where there exists an $x \in P$ such that $a^i x < b_i$, and $Q_2 \subseteq \{1, \ldots, n\}$ is the subset of variables where there exists an $x \in P$ such that $l_i < x_i < u_i$. In addition to solving the above non-linear program, finding Q_1 and Q_2 can itself be a time consuming procedure. Fortunately, commercial solvers such as CPLEX®, have powerful path following interior point methods that converge to the analytic center of the optimal face of a LP (we refer the reader to the discussion on the limiting properties of the central path in Halická (2002) and references therein). Hence, by setting the objective coefficients of the original LP relaxation to 0, we can (barring the impact of certain LP reductions in solvers such as CPLEX®) efficiently obtain a reasonable approximation of the analytic center. This is the basic strategy used in Baena and Castro (2010).

Unlike Baena and Castro (2010), Naoum-Sawaya and Elhedhli (2011) propose using a weighted analytic center where greater weight is given to constraints violated by the rounded point. They use a cutting plane approach to update their fractional points rather than projection as in FP.

The idea of giving more weight to constraints that are violated by the rounded point obtained from the analytic center seems sensible and can be incorporated into the FP setting as well. If the rounded value of an FP iterate x^* violates a particular constraint i, i.e., if

$$\sum_{j\in I}\lceil x_j^*\rfloor a_j^i + \sum_{j\in\{1,\ldots,n\}\setminus I} x_j^* a_j^i > b_i,$$

then a cut of the form

$$\sum_{j\in I} x_j a_j^i \leq b_i - \sum_{j\in\{1,\ldots,n\}\setminus I} x_j^* a_j^i,$$

is added to the linear relaxation. Note that the cut is stated only in terms of the integer variables. Naoum-Sawaya and Elhedhli propose this to avoid situations where the analytic center computed after adding the cut is different only in the continuous variables leaving the fractional components of the integer variables relatively unchanged, and thus, the rounded values also unchanged. The integer line search procedure can then be carried out with x^* as the starting point, and the new analytic center computed with all previously added cuts as the end point of the line segment. Unfortunately, this requires the analytic center to be computed each time a new fractional point x^* is obtained, which is (possibly too) expensive.

As observed earlier, the choice of the start and end points of the line segment is made to form a compromise between objective, feasibility, and integrality considerations. With the starting point x^s chosen to be the current FP iterate x^*, a good qualifier for objective and integrality considerations, the end point should be chosen to lead towards feasibility. The resulting line search creates an ordering of some subset of the integer variables that reflects the importance of the individual variables with respect to recovering feasibility from the rounded point at the start of the line segment. With this in mind, it seems unnecessary to use sophisticated interior point methods to find an end point to achieve this. The end point does not need to be feasible, it only needs to provide a direction towards feasibility. Therefore, we consider a simple scheme, which can be efficiently implemented, where a direction towards feasibility is computed using a conic combination of the constraints violated by $\lceil x^s\rfloor$. Each violated constraint is simply weighted by the extent of the violation. More precisely, given the set of constraints

$Q(x^s) \subseteq \{1, \ldots, m\}$ violated by $\lceil x^s \rfloor$, i.e.,

$$Q(x^s) = \{i \in \{1, \ldots, m\} : a^i \lceil x^s \rfloor > b_i\},$$

the direction $\bar{d}$ is computed as follows:

$$\bar{d} = \sum_{i \in Q(x^s)} \left(\frac{b_i - a^i \lceil x^s \rfloor}{\|a^i\|_2} \right) a^i.$$

Note that $(b_i - a^i \lceil x^s \rfloor)/\|a^i\|_2$ is the (signed) distance from $\lceil x^s \rfloor$ to the hyperplane $\{x : a^i x = b_i\}$. However, $\lceil x^s \rfloor + \bar{d}$ is not guaranteed to be feasible with respect to constraints $Q(x^s)$, let alone be feasible with respect to all constraints as guaranteed by interior points. Despite this, the computational results demonstrate that this simple approach is almost as effective as the more elaborate interior point schemes in finding a good direction with the added advantage of requiring considerably less computational effort. Note that if $\|a^i\|_2^2$ is used instead of $\|a^i\|_2$, then d can be interpreted as a steepest descent direction using a quadratic penalty for infeasibility. Computational experiments with this direction did not result in performance improvements.

3.3 Extending the Line Search

The end points described previously provide a direction from $\lceil x^s \rfloor$ towards LP feasibility. For the line search procedure described in Algorithm 2, this translates to providing a sequence of breakpoints and associated variables whose importance with respect to recovering feasibility from the initial rounded solution is given by the ranking of the breakpoints in the sequence.

If the primary value of the two end points x^s and x^t is providing a direction, then there is no need to restrict the search for breakpoints to the line segment defined by x^s and x^t. The line segment extended past x^t and/or before x^s can potentially give additional breakpoints. To see this, consider the example given in Figure 1 of a binary program with three variables where $x^s = (0.1, 0.1, 0.1)$ and $x^t = (0.2, 0.3, 0.4)$. In this case, all the rounded points along the line segment produce the same integer solution, i.e., $(0, 0, 0)$. Indeed, there are no breakpoints along this line. However, consider the point $\bar{x}^t = (0.3, 0.5, 0.7)$ obtained by starting at x^s and moving in the direction of x^t past x^t. The line segment connecting x^s and $\bar{x}^t$ clearly has two breakpoints and explores integer solutions $(0, 0, 1)$ and $(0,\ 1,\ 1)$ that cannot be obtained by rounding points along the line segment between x^s and x^t only. Note that $\bar{x}^t = x^s + \alpha(x^t - x^s)$ when $\alpha = 2$. The line segment can be extended beyond x^t for any value of $\alpha \geq 1$. Similarly, one can also extend the line search beyond x^s by choosing $\alpha \leq 0$.

Note too that for $\alpha > 3$, the variable bounds of at least one of the variables are violated and hence, an integer feasible solution is not obtainable for $\alpha > 3$ when extending the line in a conventional way. However, by projecting the line back onto the hypercube defined by the variable bounds (in this case the unit hypercube), it is possible to find a third integer point $(1, 1, 1)$ for $\alpha \geq 4$ (see Figure 1).

The above example demonstrates the benefits of extending the line segment on either side of x^s and x^t and projecting it back on the hypercube defined by the variable bounds. Fortunately, the breakpoints associated with such a projection can also be obtained efficiently. Given start and end point x^s and x^t respectively, possibly obtained by extending the line segment provided by some initial choice of start and end points, we filter those breakpoints that satisfy the variable bounds for the individual variables, i.e, we only consider a breakpoint $\lambda \in \Lambda_i(x_i^s, x_i^t)$ for variable i if

$$l_i \leq (1 - \lambda)x_i^s + \lambda x_i^t \leq u_i.$$

This is done efficiently as follows. As before, we define $\bar{\lambda}_i$ to be the closest breakpoint to x_i^s in the direction of the line search however, this time, we also incorporate variable bound information as follows:

$$\bar{\lambda}_i = \begin{cases} \frac{\lceil \max\{x_i^s, l_i\} \rfloor + 0.5 - x_i^s}{x_i^t - x_i^s}, & \text{if } x_i^s < x_i^t \\ \frac{\lceil \min\{x_i^s, u_i\} \rfloor - 0.5 - x_i^s}{x_i^t - x_i^s}, & \text{if } x_i^s > x_i^t. \end{cases}$$

Note that our choice of $\bar{\lambda}_i$ ensures that the rounded value of the point along the line associated with $\bar{\lambda}_i$ is feasible for l_i if $x_i^s < x_i^t$, and feasible for u_i if $x_i^s > x_i^t$. Thus, starting with $\bar{\lambda}_i$, the breakpoints for variable i have to be chosen so that the associated rounded value does not exceed u_i if $x_i^s < x_i^t$, and does not go below l_i if $x_i^s > x_i^t$. In terms of the initial breakpoint $\bar{\lambda}_i$, the set $\bar{\Lambda}_i(x^s, x^t)$ of breakpoints for variable i that are feasible with respect to the variable bounds can then be given by

$$\bar{\Lambda}_i(x^s, x^t) = \left\{ 0 < \lambda \leq 1 : \lambda = \bar{\lambda} + \frac{k}{x_i^t - x_i^s},\ k \text{ integer, and } k^- \leq k \leq k^+ \right\}$$

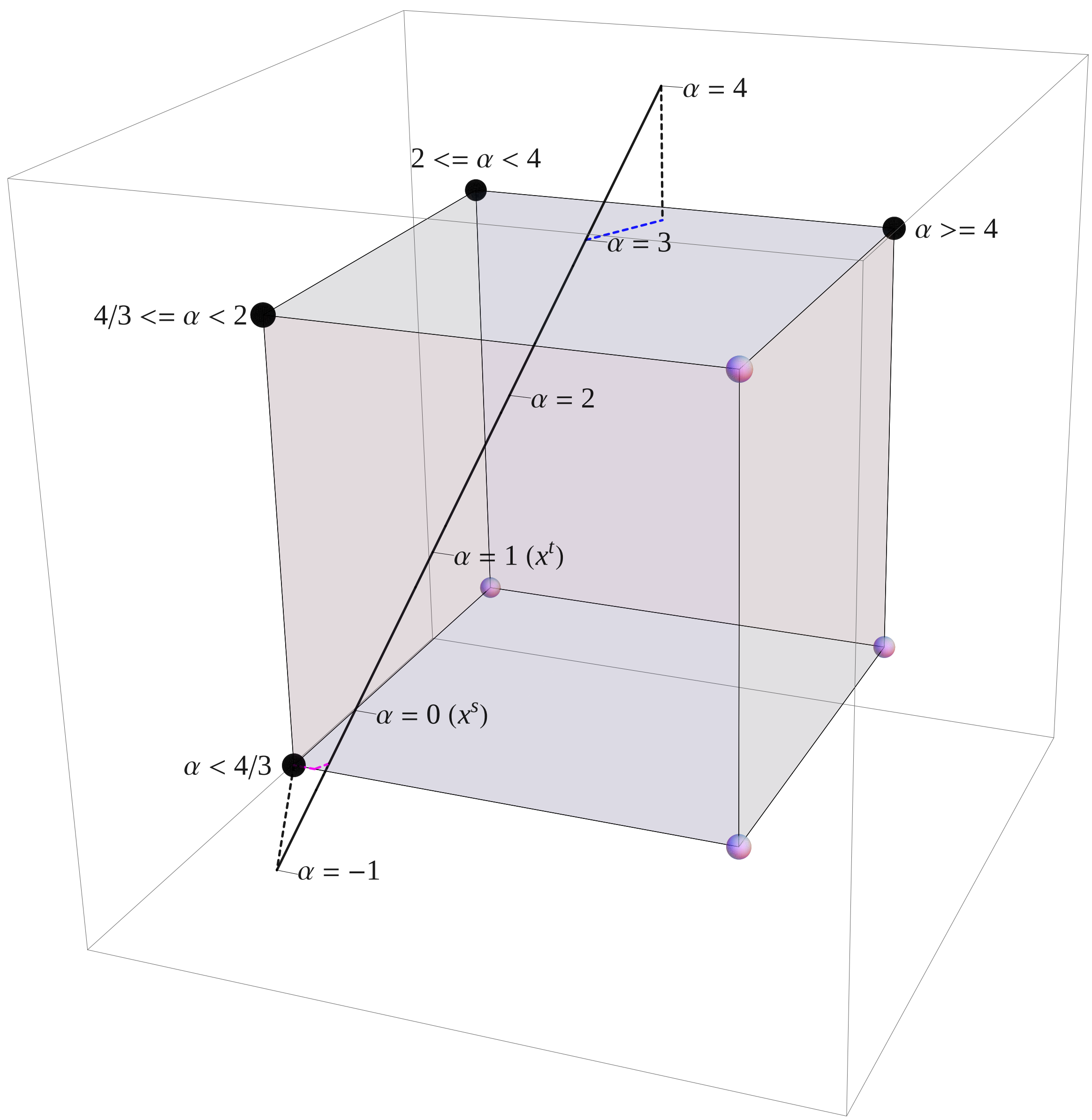

Figure 1: Roundings resulting from extending the line beyond x^s and x^t

where

$$k^- = \begin{cases} \lceil l_i - (\lceil \min\{x_i^s, u_i\} \rfloor + 1) \rceil, & \text{if } x_i^s > x_i^t \text{ and} \\ 0, & \text{otherwise} \end{cases}$$

and

$$k^+ = \begin{cases} \lfloor u_i - (\lceil \max\{x_i^s, l_i\} \rfloor + 1) \rfloor, & \text{if } x_i^s < x_i^t \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

Note that when using $\bar{\Lambda}_i(x^s, x^t)$ instead of $\Lambda_i(x^s, x^t)$ for all $i \in I$ to compute $\Psi(x^s, x^t)$, the resulting line search corresponds to rounding all points along a line segment projected back onto the hypercube defined by the variable bounds for parts of the line segment that extend beyond the boundaries of the hypercube. Indeed, a breakpoint for a variable is considered in $\bar{\Lambda}_i(x^s, x^t)$ only if it is feasible for the variable bounds of variable i. The breakpoints in $\bar{\Lambda}_i$ may be infeasible for the variable bounds of some other variable, i.e., for some $\lambda \in \bar{\Lambda}_i(x^s, x^t)$, $(1 - \lambda)x_j^s + \lambda x_j^t$ may not satisfy the variable bounds of some other variable $j \neq i$. However, Algorithm 2 only increments/decrements the value of an integer variable i if the corresponding breakpoint is in $\bar{\Lambda}_i(x^s, x^t)$ and thus, all the integer points explored by the algorithm will always satisfy the variable bound constraints. In this case, the number of breakpoints in $\Psi(x^s, x^t)$ (and thus the number of iterations of Algorithm 2) is pseudopolynomial in $|u - l|$.

3.4 Propagation within the Line Search

Constraint propagation is a general concept that refers to using inference techniques to eliminate from contention certain values for a variable (Schulte and Stuckey 2004). This can be done as a preprocessing phase to tighten the problem formulation (Savelsbergh 1994), and/or done progressively during the process of solving a problem e.g., during branch-and-bound and/or constraint programming (Achterberg 2007).

In FP, propagation techniques can be used to improve rounding. When a particular variable is rounded and fixed at that value, the impact of this fixing is propagated to reduce the domain of other variables before continuing to round the remaining variables. If D_i is the current domain for variable $i \in I$, then the rounding procedure is modified to find the closest integer point to x_i within D_i rather using naive rounding to simply find the closest integer point to x_i. Here we use

$$\lceil x_i \rfloor^D = \arg\min_{z \in D_i \cap \mathbb{Z}} |z - x_i|.$$

to denote the closest integer to x_i in D_i. The propagation algorithm itself is described in detail in Fischetti and Salvagnin (2009) and is based upon the constraint propagation systems of Rossi et al. (2006), Schulte (2000), and Schulte and Stuckey (2004). Algorithm 3 summarizes the propagation scheme (`propRound()`) used in Fischetti and Salvagnin (2009).

Input : x
Initialize: $D_i \leftarrow [l_i, u_i]$ for all $i = 1, \ldots, n$;

3.1 `/* rank integer variables */`
3.2 $\{i_1, i_2, \ldots, i_{|I|}\} \leftarrow$ `rank`(x)
3.3 `/* round integer variables in given order */`
3.4 **forall** $k = 1, \ldots, |I|$ **do**
3.5 $\quad x_{i_k} \leftarrow \lceil x_{i_k} \rfloor^D$
3.6 $\quad D \leftarrow$ `propagate`(x_{i_k})
3.7 **end**
3.8 **if** x *is an* ***incumbent*** **then**
3.9 $\quad$ **record** x;
3.10 **end**

Algorithm 3: `propRound`(x)

Here, `rank`() creates an ordering that determines the sequence in which the integer variables are rounded and propagated. This ordering is based on the fractionality of the integer variables in x. We refer the reader to Fischetti and Salvagnin (2009) for details. Note that each time a variable is rounded, the rounded value is propagated using `propagate()` to tighten the domain of other variables before continuing on to round the next integer variable. At the end, the resulting point is checked for feasibility and recorded if it is an incumbent solution. Note that in the presence of continuous variables, an LP is solved to obtain the values of these variables between Steps 3.7 and 3.8.

The propagation technique described above proved beneficial to FP, finding better integer solutions compared to using naive rounding, and often finding feasible solutions when naive rounding fails. We next show how propagation can also be incorporated to enhance the integer line search procedure.

A naive way of incorporating propagation within the line search would be to use `propRound`() to round each breakpoint encountered during the line search. However, this would destroy the efficiency of the line search that is based on changing only one integer value at a time. Instead, we use `propRound`(x^s) to determine the initial rounded point, and then proceed as before, changing the integer values of variable one at a time within the current domain of the variable, and propagate the impact of fixing a variable only when there are no further changes to be made for that variable during the line search. Algorithm 4 summarizes this procedure. Note that in the presence of continuous variables, an LP is solved to obtain the value of these variables between Steps 4.2 and 4.3.

Input : x^s and x^t
Initialize: $x \leftarrow$ `propRound`(x^s);
compute breakpoints $\Psi(x^s, x^t) = \{(i_k, \lambda_k, d_k)_{k=1,\ldots,K}\}$;

4.1 **forall** $k = 1, \ldots, K$ **do**
4.2 $\quad x_{i_k} \leftarrow \lceil x_{i_k} + d_k \rfloor^D$
4.3 $\quad$ **if** *x is an **incumbent*** **then**
4.4 $\quad\quad$ **record** x;
4.5 $\quad$ **end**
4.6 $\quad$ **if** *no more changes for i_k* **then**
4.7 $\quad\quad D \leftarrow$ `propagate`(x_{i_k});
4.8 $\quad$ **end**
4.9 **end**

Algorithm 4: The integer line search procedure with propagation.

Algorithm 4 not only maintains the advantage of exploring many integer points and doing this by only changing one variable at a time, but additionally, the number of calls to `propagate()` is at most twice the number of integer variables, i.e., at most once for each integer variable when finding the initial rounded solution, and at most once for each integer variable during the line search it self.

An important side benefit of propagation is that fewer integer variables are changed in value, as we only round to an integer point within the allowable domain that is constantly shrinking, and thus fewer LP solves are required to find the values of the continuous variables. As solving LPs, even when using warm starts, is the most computationally intensive part of the line search, incorporating propagation often leads to a reduction in computing time.

4 A Computational Study

In this section, we present the results of a comprehensive computational study that investigates our proposed enhancements to FP. The primary goal of the study is to assess the benefits, if any, of replacing the simple rounding step of the original FP with our more effective exploration of rounded solutions along a line segment. The secondary goal of the study is to decipher the merits of some of the ideas underpinning the integer line search, e.g., the choice of end points, the extension of the line search, projecting the line back onto the hypercube defined by the variable bounds, and incorporating propagation within the line search.

All experiments were conducted on 3.16 GHz Intel® Xeon® processors with 64 GB of RAM (with a limit of 2GB per process), and CPLEX® is used for solving the various LP relaxations and projection problems, and for approximating the analytic center using its path following "Barrier" algorithm.

An extensive test-bed of 1304 instances from MIPLIB2003[1], MIPLIB2010[2], COR@L[3], and OR-LIB[4] formed the basis for our experiments. We eliminate 31 instances from the test set when reporting results: 10 instances because the LP relaxation could not be solved within 30 minutes, 2 instances because they had an infeasible LP, and 19 instances because they exhausted available memory in one of our experiments.

In our computational study, we evaluate and compare the performance of the following variants of FP: the original FP, denoted by FP, the original FP with constraint propagation (Fischetti and Salvagnin (2009)), denote by FP^{+}, and the original FP with our integer line search, denoted by $FP^{+}_{ls(e,[s,t],p)}$, where e indicates the end point used, either c for conic, a for analytic center, or $\bar{a}$ for weighted analytic center, $[s,t]$ indicates the search interval (i.e., choice of α for extending the line search), either [0,1] or [-1,2], p indicates whether the line is projected back onto the hypercube defined by the variable bounds or not, either y or n, and superscript $^{+}$ denotes that

[1] `http://miplib.zib.de/miplib2003/`
[2] `http://miplib.zib.de/miplib2010/`
[3] `http://coral.ie.lehigh.edu/~mip-instances/instances/`
[4] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mipinfo.html`

constraint propagation is employed during the search (both during FP and in the line search). The evaluation and comparison is based on executing FP once starting from a solution to the LP relaxation of an instance.

The variants of FP with integer line search are incorporated in the C++ implementation of FP that was kindly provided by Fischetti and Salvagnin. This implementation of FP also gives us access to FP^+, as constraint propagation can be activated simply by setting the appropriate option. The variants of the integer line search are incorporated in such a way that whenever a rounding step is executed in FP with an LP feasible but fractional point x^*, the integer line search is also executed with x^* as the start point of the line segment. However, what happens during the integer line search is only recorded for analysis purposes. After the integer line search has been executed, FP proceeds as if the execution of the integer line search did not take place. This allows for a fair comparison between the different line searches as the FP trajectory is not altered. In all experiments, a 30 minute time limit is imposed on the total time taken by FP, including the time taken by the integer line searches.

As mentioned above, the primary goal of the study is to assess the benefits, if any, of replacing the simple rounding step of the original FP with a more involved exploration of rounded solutions along a line segment. We start by reporting on the performance of FP^+, and $FP^+_{ls(c,[-1,2],y)}$ compared to FP. We shall later assess the impact of the various algorithmic choices within the line search discussed in §3.3.

Figure 2 shows a performance profile of the percentage improvement in solution quality produced by FP^+ and $FP^+_{ls(c,[-1,2],y)}$ relative FP. (The performance profile for FP^+ is created as follows. For a given instance, let $v(FP^+)$ and $v(FP)$ denote the value of the feasible solution produced by FP^+ and FP, respectively. For instances where FP^+ produced a better solution than FP, the relative percentage improvement in solution quality is given by $100 \times \frac{v(FP)-v(FP^+)}{|v(FP)|}$. The performance profile shows, for a given level l of relative percentage improvement in solution quality, the percentage of instances in the test set with a relative percentage improvement in solution quality of at least l. For example, for FP^+ about 26% of the instances in the test set have a relative percentage improvement in solution quality of at least 10%. The part of the performance profile beyond 100% is used to indicate the percentage of instances where FP^+ found a feasible solution, but FP did not. The performance profile for $FP^+_{ls(c,[-1,2],y)}$ is constructed analogously.)

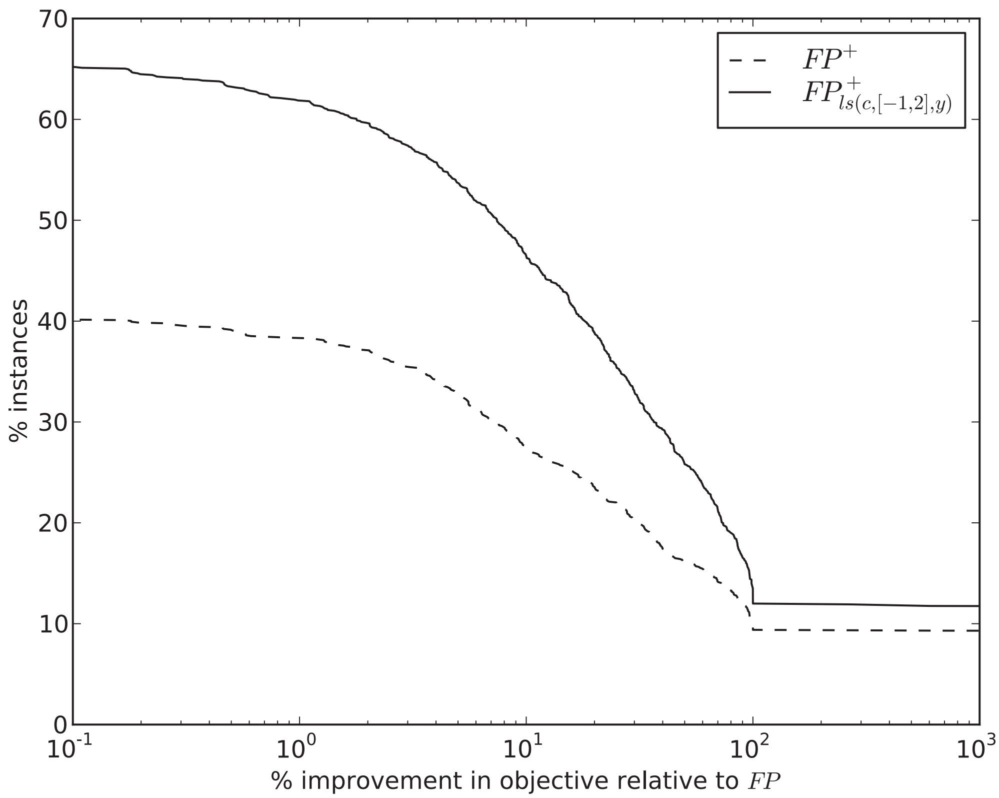

Figure 2: Performance profile showing improvement in solution quality relative to FP

Figure 2 clearly demonstrates that significant gains result from incorporating the integer line search. We first observe that FP^+ finds a solution of higher quality than FP for about 40% of the instances, and FP^+ finds feasible solutions for about 9% of the instances where FP failed to find a feasible solution. Moreover, $FP^+_{ls(c,[-1,2],y)}$ finds a solution of higher quality than FP for about 66% of the instances, and $FP^+_{ls(c,[-1,2],y)}$ finds feasible solutions for about 12% of the instances where FP failed to find a feasible solution. This not only represents a significant improvement over FP, but also a noticeable improvement over FP^+. Of course, since the FP iterates in FP are different to the iterates in FP^+ (and thus $FP^+_{ls(c,[-1,2],y)}$ since the statistics for $FP^+_{ls(c,[-1,2],y)}$ were collected during the execution of FP^+), it is not guaranteed that FP^+ and $FP^+_{ls(c,[-1,2],y)}$ produce solutions that are no worse than that of FP.

Figure 3 shows a performance profile of the percentage deterioration in solution quality produced by FP^+, and $FP^+_{ls(c,[-1,2],y)}$ relative to FP. From Figure 3 we observe that FP^+ produces a solution of worse quality than FP in about 13% of instances whereas $FP^+_{ls(c,[-1,2],y)}$ produces a solution of worse quality in 9% of instances.

Even though we have purposely and carefully designed the integer line search to be as efficient as possible, it does require additional computations. Figure 4 shows a performance profile of the cumulative time taken to solve instances for each of FP, FP^+, and $FP^+_{ls(c,[-1,2],y)}$. Here, the total time over all instances is reported in parenthesis within the figure's legend. Not surprisingly, the variant of FP that incorporates the integer line search is less efficient then the others, but the loss in efficiency is relatively small and well worth it given the gains in solution quality. $FP^+_{ls(c,[-1,2],y)}$ is on average about 1.22 times slower than FP^+. We note here too that the integer line search without propagation, i.e., $FP_{ls(c,[-1,2],y)}$, required 85,037 seconds over all instances compared to 57,055 seconds for the integer line search with propagation, i.e., $FP^+_{ls(c,[-1,2],y)}$. This somewhat surprising and maybe counterintuitive observation is explained by the fact that propagation leads to fewer changes in values for the integer variables, resulting in fewer LP solves for the continuous variables.

Next, we focus on analyzing some of our algorithmic choices. Because the start point of the line segment is the current FP iterate x^* and therefore should be a good qualifier for objective and integrality considerations, the end point is chosen to lead towards feasibility. For efficiency reasons, we have

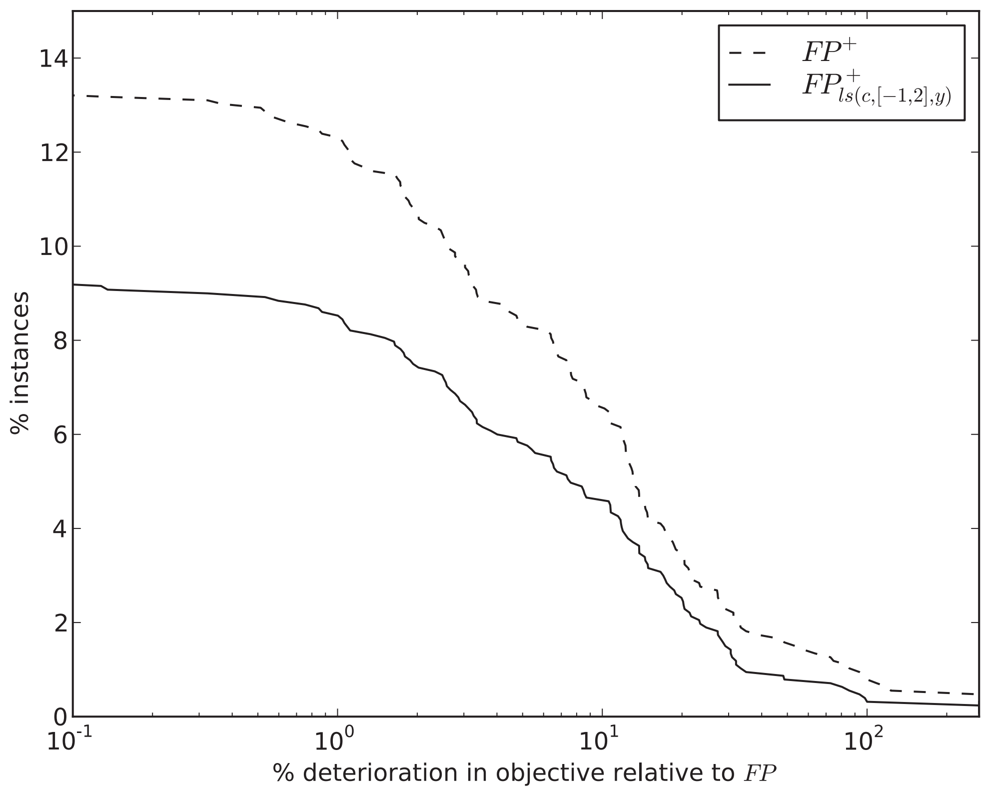

Figure 3: Performance profile showing deterioration in solution quality relative to FP

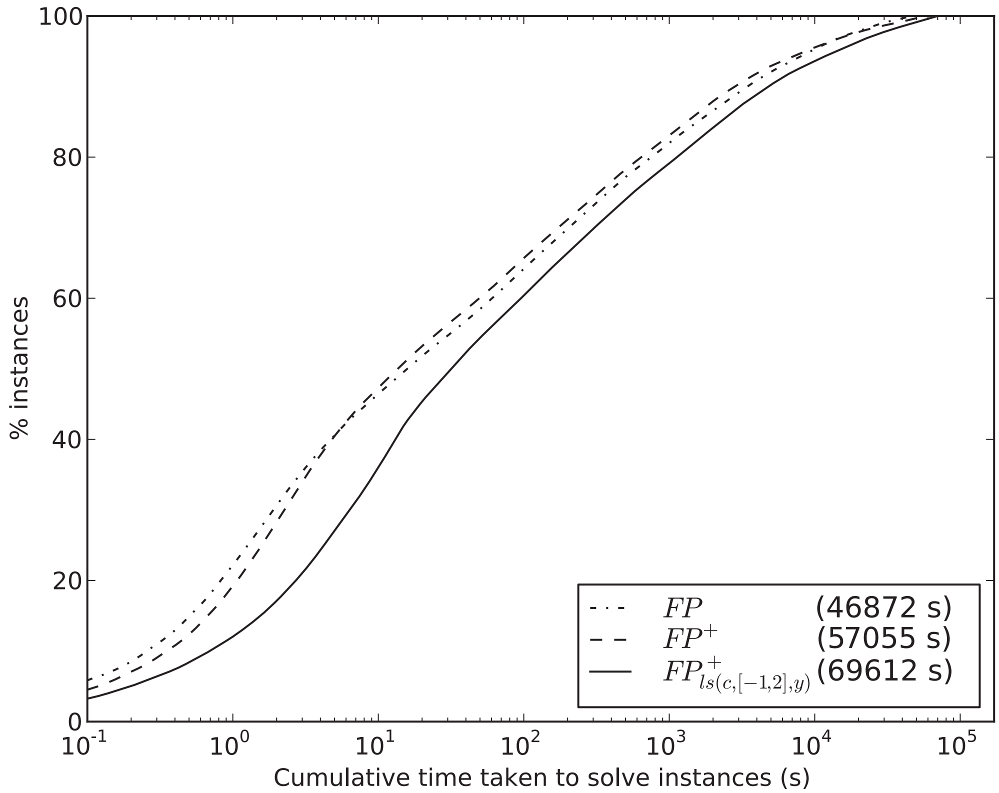

Figure 4: Performance profile showing the cumulative time taken to solve instances

chosen to use a simple scheme using a conic combination of the constraints violated by $\lceil x^* \rfloor$. We investigate the merit of this choice by comparing it to using the analytic center and the weighted analytic center. Figure 5 shows a performance profile of the percentage improvement in solution quality produced by $FP^+_{ls(c,[-1,2],y)}$, $FP^+_{ls(a,[-1,2],y)}$, and $FP^+_{ls(\bar{a},[-1,2],y)}$ over FP^+. We see that using a weighted analytic center produces the best results. Doing so produces a solution of higher quality for 46% of the instances, whereas using the conic combination results in a solution of higher quality in 42% of the instances. There is hardly any difference when it comes to finding feasible solution where FP^+ failed to do so; regardless of the end point used, for about 3% of the instances where FP^+ fails to find a feasible solution, the line search variants are successful.

The superiority in solution quality of $FP^+_{ls(\bar{a},[-1,2],y)}$ over $FP^+_{ls(c,[-1,2],y)}$ (however marginal) is not necessarily the result of the choice of end point and thus of the search direction. Indeed, Figure 6, which shows a performance profile of the average number of breakpoints for each integer line search, indicates clearly that $FP^+_{ls(a,[-1,2],y)}$ and $FP^+_{ls(\bar{a},[-1,2],y)}$ explore significantly more breakpoints, and thus integer points, than $FP^+_{ls(c,[-1,2],y)}$. To understand the impact of the number of breakpoints on the observed differences in performance, we reran the experiment, but limited the number of breakpoints explored by $FP^+_{ls(a,[-1,2],y)}$ and $FP^+_{ls(\bar{a},[-1,2],y)}$ to the number of breakpoints explored by $FP^+_{ls(c,[-1,2],y)}$. In Figure 7, we show results of this experiment. We see that the performance of the three integer line searches is comparable and that it might even be argued that the performance of $FP^+_{ls(c,[-1,2],y)}$ is slightly better. Thus, the performance difference observed earlier is not caused by the choice of end points, but by the difference in number of rounded solutions explored.

This raises the question as to whether the search direction is important at all. Therefore, we also conducted an integer line search where the direction is chosen randomly as follows. Given x^*, a direction for the i^{th} variable is chosen randomly between $x_i^* - l_i$ and $u_i - x_i^*$. Again, the number of breakpoints explored is limited by the number explored by $FP^+_{ls(c,[-1,2],y)}$. This "random" variant of the line search, denoted by $FP^+_{ls(r,[-1,2],y)}$, is compared to $FP^+_{ls(c,[-1,2],y)}$ in the performance profile shown in Figure 8. We observe that the random direction, where the only consideration towards feasibility is made with respect to the variable bounds, finds a better solution than FP^+ for 36% of the instances. Hence, a large proportion of the gains of the integer line searches over FP^+ can be attributed simply to exploring a large number of integer points around x^*, which was the original motivation for our research. That said, the results also demonstrate that there is clearly an advantage to exploring integer points along a line moving towards LP feasibility as $F^+_{ls(c,[-1,2],y)}$ finds a solution of higher quality than $FP^+_{ls(r,[-1,2],y)}$ for 8% more of the instances.

We have shown that a direction computed simply as a conic combination of violated constraints produces equally good solutions (and often outperforms) directions computed using more elaborate analytic centers. However, the main benefit is that $FP^+_{ls(a,[-1,2],y)}$ and $FP^+_{ls(\bar{a},[-1,2],y)}$ are on average 2.5 times slower than FP^+ whereas $FP^+_{ls(c,[-1,2],y)}$ is only 1.22 times slower than FP^+. The low computational burden of $FP^+_{ls(c,[-1,2],y)}$ is the result of the fact that the conic direction can be computed efficiently and that considerably fewer breakpoints are explored as analytic centers typically have a much larger number of non-zero values compared to the number of variables that contribute to the constrains violated by the starting rounded solution. This is turn results in considerably fewer LP solves for finding the values of the continuous variables.

In §3.3, we have shown, by means of an example, that extending the search beyond the end points of the line segment and projecting the line back onto the hypercube defined by the variable bounds can result in finding additional integer points. In the next experiment, we assess the impact of each of these features by comparing FP^+, $FP^+_{ls(c,[0,1],y)}$, $FP^+_{ls(c,[-1,2],n)}$, and $FP^+_{ls(c,[-1,2],y)}$. The performance profiles are shown in Figure 9. We see that projecting the line back onto the hypercube defined by the variable bounds has the most noticeable impact on improving solution quality while extending the line search beyond the start and end points has less of an impact. It is worth noting that the impact of extending and projecting the line search is much more pronounced when propagation is turned off. In fact, extending and projecting the line search provides, in some sense, a form of propagation as it prevents changes in values for integer variables outside their bounds.

5 Final Remarks

We have shown that replacing the rounding step of the feasibility pump with an integer line search can significantly enhance its performance at modest computational cost. The success is the result of a combination of innovative ideas and clever engineering. Our work represents a small incremental step forward in our quest to solve larger and more difficult integer programs better and faster. There are still more ideas that can be explored to improved the efficiency of our proposed approach. For example, for instances with a large number of breakpoints and continuous variables, the computational burden of having to solve a LP to obtain the value of the continuous variables, even when exploiting warm starts, is substantial. In such situations, it may be worth solving for the continuous variables only for a limited number of breakpoints suitably chosen in the range of possible breakpoints. Another idea is to search in several directions or in a "tree-like" manner to potentially explore even more rounded solutions near the start point of the search. We have left these ideas for future research as they are mostly "variations on a theme" and key ideas are already present, computationally tested, and discussed in the current paper.

Acknowledgements

We thank Domenico Salvagnin for providing the source code of his implementation of the feasibility pump with constraint propagation and for answering all of our questions promptly, elaborately, and clearly.

References

Achterberg, T. 2007. Constraint integer programming. Ph.D. thesis, TU Berlin.

Achterberg, T., T. Berthold. 2007. Improving the feasibility pump. *Discrete Optimization* **4**(1) 77–86.

Baena, D., J. Castro. 2010. Using the analytic center in the feasibility pump. *Optimization Online* URL `www.optimization-online.org/DB_HTML/2010/11/2793.html`. Last accessed 24/05/11.

Balas, E., C.H. Martin. 1980. Pivot and Complement-A Heuristic for 0-1 Programming. *Management Science* **26**(1) 86–96.

Balas, E., S. Schmieta, C Wallace. 2004. Pivot and shift-a mixed integer programming heuristic. *Discrete Optimization* **1**(1) 3–12.

Bertacco, L., M. Fischetti, A. Lodi. 2007. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization* **4**(1) 63–76.

Bixby, R.E. 2002. Solving Real-World Linear Programs: A Decade and More of Progress. *Operations Research* **50**(1) 3–15.

Bixby, R.E., E. Rothberg. 2007. Progress in computational mixed integer programming - A look back from the other side of the tipping point. *Annals of Operations Research* **149** 37–41.

Danna, E., E. Rothberg, C. Le Pape. 2005. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming* **102** 71–90.

De Santis, M., S. Lucidi, F. Rinaldi. 2010. New concave penalty functions for improving the feasibility pump. *Optimization Online* URL `www.optimization-online.org/DB_HTML/2010/07/2667.html`. Last accessed 24/05/11.

Eckstein, J., M. Nediak. 2007. Pivot, Cut, and Dive: a heuristic for 0-1 mixed integer programming. *Journal of Heuristics* **13** 471–503.

Faaland, B.H., F.S. Hillier. 1979. Interior Path Methods for Heuristic Integer Programming Procedures. *Operations Research* **27**(6) 1069–1087.

Fischetti, M., F. Glover, A. Lodi. 2005. The feasibility pump. *Mathematical Programming* **104** 91–104.

Fischetti, M., A. Lodi. 2003. Local branching. *Mathematical Programming* **98** 23–47.

Fischetti, M., D. Salvagnin. 2009. Feasibility pump 2.0. *Mathematical Programming Computation* **1** 201–222.

Glover, F., M. Laguna. 1997a. General purpose heuristics for integer programmingPart I. *Journal of Heuristics* **2** 343–358.

Glover, F., M. Laguna. 1997b. General purpose heuristics for integer programmingPart II. *Journal of Heuristics* **3** 161–179.

Halická, M. 2002. Analyticity of the central path at the boundary point in semidefinite programming. *European Journal of Operational Research* **143**(2) 311 – 324.

Hanafi, S., J. Lazic, N. Mladenovic. 2010. Variable Neighbourhood Pump Heuristic for 0-1 Mixed Integer Programming Feasibility. *Electronic Notes in Discrete Mathematics* **36** 759–766.

Hillier, F.S. 1969. Efficient Heuristic Procedures for Integer Linear Programming with an Interior. *Operations Research* **17**(4) 600–637.

Jeroslow, R.G., T.H.C. Smith. 1975. Experimental results on Hillier's linear search. *Mathematical Programming* **9** 371–376.

Lokketangen, A., F Glover. 1998. Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research* **106**(2-3) 624–658.

Naoum-Sawaya, J., S. Elhedhli. 2011. An interior point cutting plane heuristic for mixed integer programming. *Computers and Operations Research* **38**(9) 1335–1341.

Rossi, F., P. van Beek, T. Walsh. 2006. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA.

Savelsbergh, M. W. P. 1994. Preprocessing and probing techniques for mixed integer programming problems. *Informs Journal on Computing* **6**(4) 445–454.

Schulte, C. 2000. Programming constraint services. Ph.D. thesis, Universität des Saarlandes, Naturwissenschaftlich-Technischen Fakultät I, Saarbrücken, Germany.

Schulte, C., P.J. Stuckey. 2004. Speeding up constraint propagation. M. Wallace, ed., *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference*. Lecture Notes in Computer Science, Springer, 619–633.

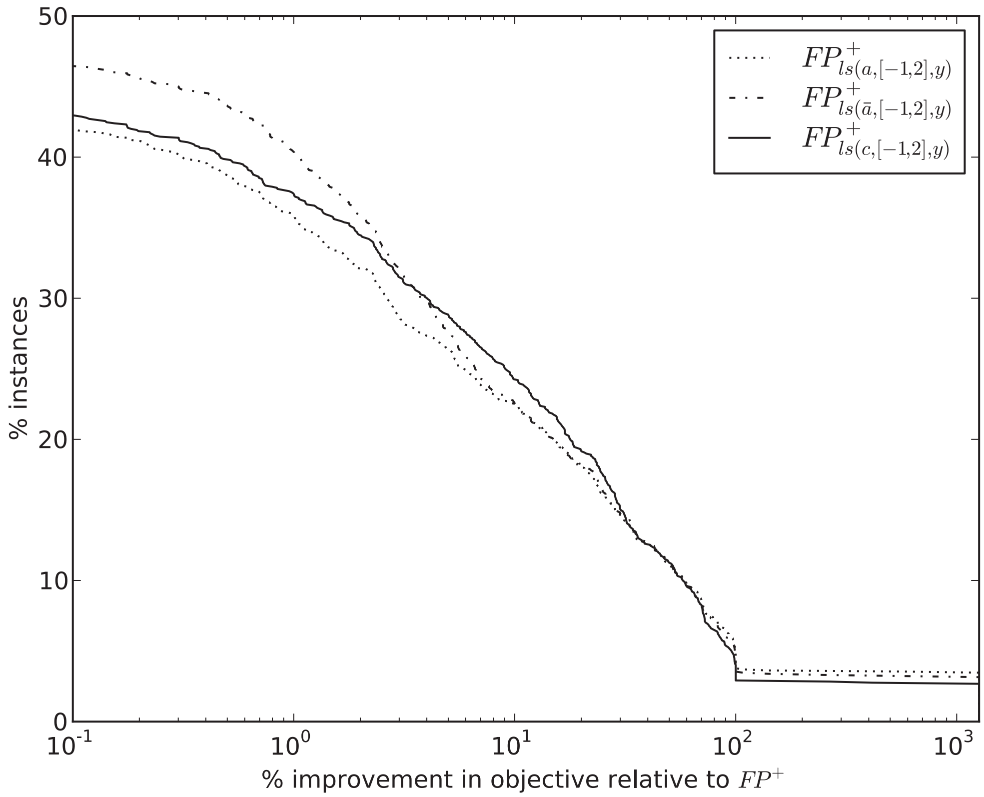

Figure 5: Performance profile showing improvement in solution quality relative to FP^+

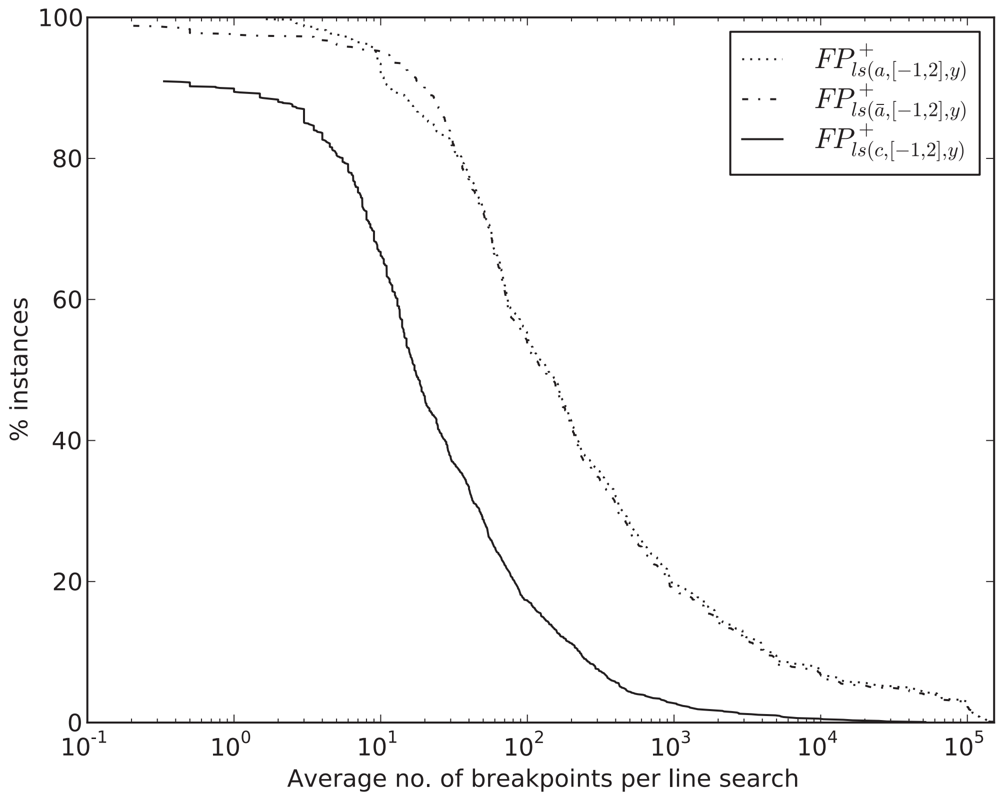

Figure 6: Performance profile showing average number of breakpoints explored during the line search

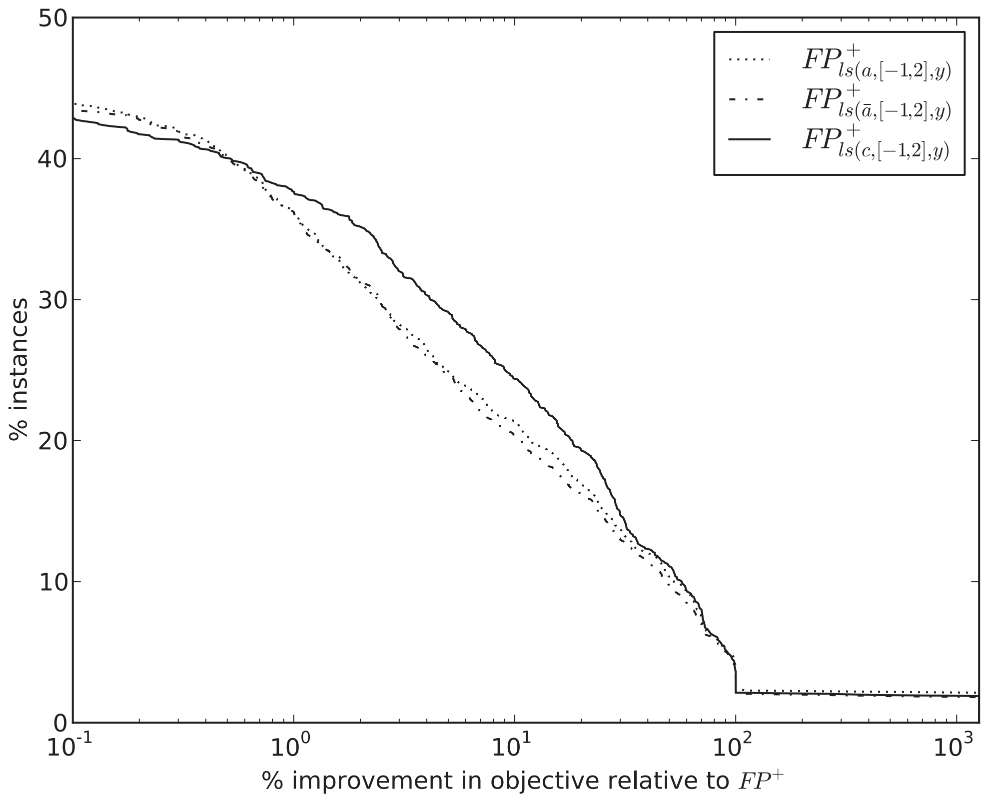

Figure 7: Performance profile showing improvement in solution quality relative to FP^+ when limiting the number of breakpoints to that explored by $FP^+_{ls(c,[-1,2],y)}$

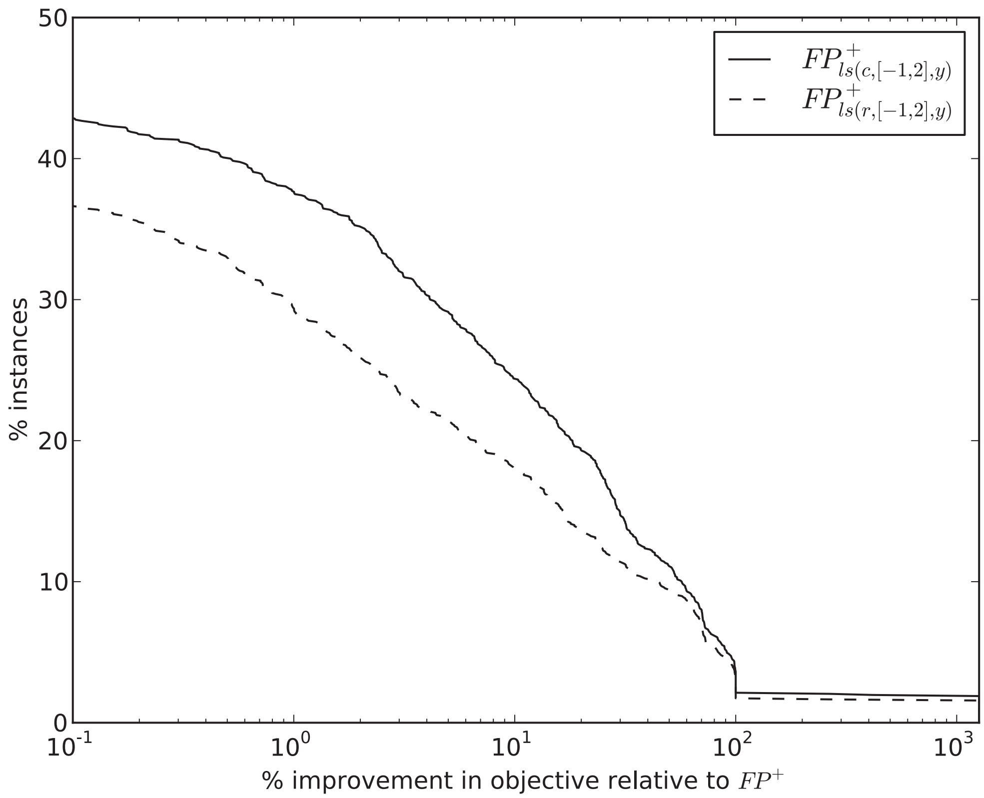

Figure 8: Performance profile showing improvement in solution quality relative to FP^+ when limiting the number of breakpoints to that explored by $FP^+_{ls(c,[-1,2],y)}$

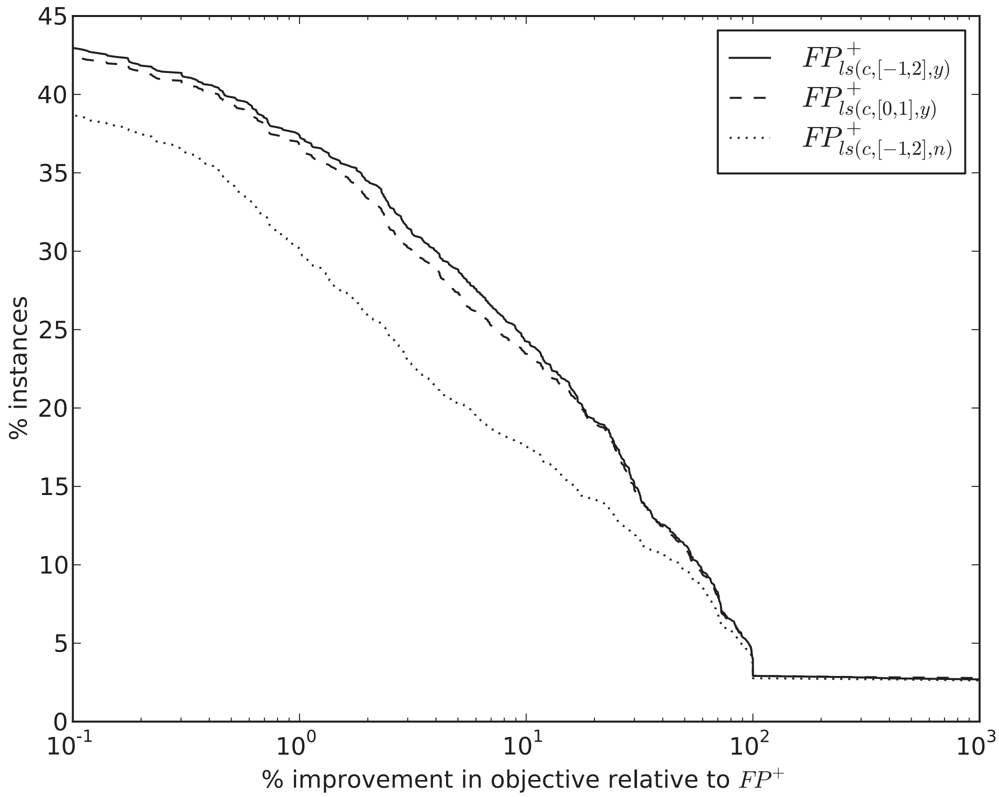

Figure 9: Performance profile showing improvement in solution quality relative to FP^+