Zuse Institute Berlin

ZUSE INSTITUTE BERLIN

Takustr. 7
14195 Berlin
Germany

TIMO BERTHOLD[1], GIONI MEXI[2], DOMENICO SALVAGNIN[3]

Using Multiple Reference Vectors and Objective Scaling in the Feasibility Pump

[1]Fair Isaac Germany GmbH,Stubenwald-Allee 19,64625 Bensheim, Germany
[2]Zuse Institute Berlin,Takustraße 7, 14195 Berlin, Germany
[3]Department of Information Engineering, University of Padova, Via Gradenigo 6/b, 35131, Padova, Italy

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30 84185-0
Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Using Multiple Reference Vectors and Objective Scaling in the Feasibility Pump

Timo Berthold[1], Gioni Mexi[2], Domenico Salvagnin[3]

July 26, 2022

Abstract

The Feasibility Pump (FP) is one of the best-known primal heuristics for mixed-integer programming (MIP): more than 15 papers suggested various modifications of all of its steps. So far, no variant considered information across multiple iterations, but all instead maintained the principle to optimize towards a single reference integer point. In this paper, we evaluate the usage of multiple reference vectors in all stages of the FP algorithm. In particular, we use LP-feasible vectors obtained during the main loop to tighten the variable domains before entering the computationally expensive enumeration stage. Moreover, we consider multiple integer reference vectors to explore further optimizing directions and introduce alternative objective scaling terms to balance the contributions of the distance functions and the original MIP objective.

Our computational experiments demonstrate that the new method can improve performance on general MIP test sets. In detail, our modifications provide a 29.3% solution quality improvement and 4.0% running time improvement in an embedded setting, needing 16.0% fewer iterations over a large test set of MIP instances. In addition, the method's success rate increases considerably within the first few iterations. In a standalone setting, we also observe a moderate performance improvement, which makes our version of FP suitable for the two main use-cases of the algorithm.

1 Introduction

Mixed-integer programming is widely used for modeling and solving challenging optimization problems that arise from real-world applications. We define a mixed-integer program (MIP) in matrix and vector notion as follows.

Definition 1.1. *(mixed-integer program) Let $m, n \in \mathbb{Z}_{\geq 0}$. Given some matrix $A \in \mathbb{Q}^{m \times n}$, vectors $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, $l, u \in \mathbb{Q}^n \cup \{\pm\infty\}$, and a set of indices*

[1] 0000-0002-6320-8154
[2] 0000-0003-0964-9802
[3] 0000-0002-0232-2244

$\mathcal{I} \subseteq \mathcal{N} := \{1, \ldots, n\}$, *a problem in the form*

$$\begin{aligned} \underset{x \in \mathbb{Q}^n}{min} \quad & c^\top x \\ s.t. \quad & Ax \leq b \\ & l \leq x \leq u \\ & x_j \in \mathbb{Z} \quad \textit{for all } j \in \mathcal{I}, \end{aligned} \tag{1}$$

is called a mixed integer program (MIP).

Finding a feasible solution to a given mixed integer programming (MIP) model is a very important, $\mathcal{NP}$-complete problem that can be extremely hard in practice and is therefore often approached via primal heuristics [1, 2]. One of the most successful schemes for finding initial feasible solutions for MIPs is the Feasibility Pump, originally proposed by Fischetti, Glover, and Lodi [3].

The fundamental idea of all FP algorithms is to construct two sequences of points that hopefully converge to a feasible solution to a given optimization problem; in our case, a MIP. One sequence consists of points that are feasible for a continuous relaxation (typically the LP relaxation), but possibly integer infeasible. The other sequence consists of points that are integral but might violate some of the imposed (linear) constraints. The next point of one sequence is always generated by minimizing the distance to the last point of the other sequence, using different distance measures in either case (e.g., the ℓ_1 and the ℓ_2 norm). For an overview of FP variants, we refer to [4].

The outline of this paper is as follows. In Section 2 we describe the core idea of FP, concentrating on the Objective Feasibility Pump 2.0 (OFP2), which is the baseline algorithm of our work. In Sections 3 and 4, we will describe one of our two main contributions, namely variations of all three stages of the FP algorithm to consider multiple reference vectors in each Stage. Section 5 describes our second contribution, namely the application of an alternative objective scaling method, to better balance optimality and feasibility considerations. Finally, in Sections 6 and 7 we present computational results with a detailed performance comparison in both an embedded and a standalone setting.

2 The Objective Feasibility Pump 2.0

The Feasibility Pump (FP) was originally introduced by Fischetti, Glover, and Lodi [3]. In Algorithm 1, we outline the principle idea of FP. In the main loop (called *pumping loop*) the algorithm constructs two sequences of vectors $\{\bar{x}^k\}_{k=1}^K$ and $\{\tilde{x}^k\}_{k=1}^K$ for a finite K. These sequences satisfy the feasibility requirements of the MIP in a complementary way. More precisely, the first sequence contains *LP-feasible vectors*, i.e., vectors from

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

and the second contains *integer vectors*, i.e., vectors that satisfy the integrality constraints, but are not necessarily LP-feasible. The goal of FP is to iteratively reduce the distance between vectors of the two sequences and converge to a feasible MIP solution. The distance between two vectors $\bar{x}^k$ and $\tilde{x}^k$ is defined as

$$\Delta(\bar{x}^k, \tilde{x}^k) := \sum_{j \in \mathcal{I}} \left| \bar{x}_j^k - \tilde{x}_j^k \right|.$$

In every iteration k, the vector $\tilde{x}^k$ is obtained in the *rounding step* (line 5), by rounding all values $\bar{x}_j^{k-1} \notin \mathbb{Z}$ with $j \in \mathcal{I}$ to an integer. The vector $\bar{x}^k$ is obtained in the *projection step* (line 8) by solving the optimization problem

$$\arg\min\{\Delta(x, \tilde{x}^k) : x \in P\}.$$

Linearizing $\Delta(x, \tilde{x}^k)$ requires the addition of auxiliary variables and constraints for every $\tilde{x}_j^k$, $j \in \mathcal{I}$ with $l_j < \tilde{x}_j^k < u_j$. More specifically, $\Delta(x, \tilde{x}^k)$ can be written as

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

where the auxiliary variables d_j satisfy the constraints

$$d_j \geq \tilde{x}_j^k - x_j \text{ and } d_j \quad \geq x_j - \tilde{x}_j^k.$$

Note that for binary MIPs, no auxiliary variables and constraints are required.

One major issue of FP is cycling. During the pumping loop, it can happen that the current integer vector $\tilde{x}^k$ is the same as a previous integer vector $\tilde{x}^r$. If the cycle has length one, hence $r = k - 1$, a "slight" perturbation is applied to $\tilde{x}^k$, which means that a small number of its integer components are chosen and their value is flipped. In the case of a cycle with a length greater than one, a "strong" perturbation, also known as a *restart* is applied. In general, frequent use of perturbations and restarts is considered not desirable, since they often counteract the progress made on reducing the distance of the two sequences.

Algorithm 1: Feasibility Pump (core idea)

Input : A MIP in the form (1)
Initialization: $\bar{x}^0 \leftarrow \arg\min\{c^\top x : x \in P\}$, $k \leftarrow 0$
Output : A feasible solution of the MIP or $\varnothing$ if none found

1 **while** *(not termination condition)* **do**
2 $k \leftarrow k + 1$
3 **if** $\bar{x}^{k-1}$ *is integer* **then**
4 **return** $\bar{x}^{k-1}$
5 $\tilde{x}^k \leftarrow round(\bar{x}^{k-1})$ `/* Rounding step */`
6 **if** *cycle detected* **then**
7 $\tilde{x}^k \leftarrow perturb(\tilde{x}^k)$ `/* Perturbation step */`
8 $\bar{x}^k \leftarrow \arg\min\{\Delta(x, \tilde{x}^k) : x \in P\}$ `/* Projection step */`
9 **return** $\varnothing$

The computational experiments in [3] show that FP has a satisfactory success rate for finding feasible solutions for binary MIPs. However, for MIPs with both binary and integer variables, the success rate of FP is lower, and the algorithm is more likely to cycle.

Bertacco et al. [5] proposed to split FP in three different stages. In Stage 1, the integrality constraints on the general integer variables are relaxed before the execution of the pumping loop. Hence, Stage 1 searches for a "binary" feasible solution, i.e., an LP-feasible vector satisfying the integrality constraints

on binary variables. Then, in Stage 2 the integrality constraints on general integers are restored and the pumping loop proceeds. As a last resort a so-called *enumeration stage*, also known as Stage 3, is called. The main idea of Stage 3 is to search for feasible solutions in the proximity of an "almost" feasible integer vector. Therefore, the original MIP is solved after replacing the objective function by the distance $\Delta(x, \tilde{x}^*)$, where $\tilde{x}^*$ is the integer vector closest to the polyhedron P obtained in the previous stages.

A major drawback of FP is that it considers the original objective of the MIP only in the initialization phase. Thus, the objective value of the final solution is often of low quality. Achterberg and Berthold [6] propose to replace the objective of projection step by a convex combination of the distance to a single integer reference vector $\tilde{x}^k$ and a term considering the original objective function of the MIP, i.e.,

$$\Delta_{\alpha_k}(x, \tilde{x}^k) := (1 - \alpha_k)\Delta(x, \tilde{x}^k) + \frac{\alpha_k \|\Delta\|_2}{\|c\|_2} c^\top x, \tag{2}$$

with $\alpha_k \in [0, 1]$. The idea of OFP is to dynamically adjust the influence of $\Delta(x, \tilde{x}^k)$ and $c^\top x$ in the projection step. In early iterations of the pumping loop, the projection step prioritizes solutions with good quality over feasibility, and gradually, the priority of feasibility over solution quality is increased. The aim of the Euclidean norms $\|\Delta\|_2$ and $\|c\|_2$ is to bring the terms $\Delta(x, \tilde{x}^k)$ and $c^\top x$ on the same scale. Note that $\|\Delta\|_2$ is the Euclidean norm of the objective function vector in $\Delta(x, \tilde{x}^k)$ and is equal to $\sqrt{|\mathcal{I}|}$.

Moreover, OFP incorporates an adjustment in the restart mechanism. Here, revisiting integer vectors does not necessarily mean that the algorithm stalls since the parameter α_k is dynamically adjusted. Consequently, if the current integer vector $\tilde{x}^k$ is equal to $\tilde{x}^r$ with $r \in \{1, \ldots, k-2\}$, the authors propose to apply the restart procedure only if $\alpha_l - \alpha_k \leq \delta$ with $\delta \in [0, 1]$ being a fixed threshold parameter.

Fischetti and Salvagnin [7] introduced the Feasibility Pump 2.0 (FP2.0), which integrates techniques from constraint programming in the rounding step of FP. Note that OFP and FP2.0 can be combined. The resulting algorithm is called *Objective Feasibility Pump 2.0* (OFP2) and is considered the baseline algorithm for our implementation. See the overview paper of Fischetti and Salvagnin [4] for (many) further variants of FP.

3 Multiple Reference Vectors in Stage 3

When embedded in MIP solvers, primal heuristics are usually run only for a short amount of time. Therefore, the computationally demanding Stage 3 of FP is deactivated or only used under certain circumstances, e.g., if the distance of the closest integer vector to the LP-polyhedron P is below a certain threshold. Therefore, we design an alternative Stage 3 that uses multiple reference vectors obtained from FP and the large neighborhood search heuristic RENS [8, 9].

3.1 Multi-Reference RENS

The principle idea of RENS is to search for feasible solutions for a MIP in the proximity of the optimal solution of its LP relaxation $\bar{x}^0$. More specifically,

RENS solves a sub-MIP of the original MIP with tighter bounds for the integer variables based on their value in $\bar{x}^0$. To that end, for each $j \in \mathcal{I}$, if $\bar{x}^0_j$ is an integer, then x_j is fixed at this value. Otherwise, the bounds of x_j are tightened to the two nearest integer values of $\bar{x}^0_j$, i.e., l_j is set to $\lfloor \bar{x}^0_j \rfloor$, and u_j to $\lceil \bar{x}^0_j \rceil$. To summarize RENS solves the sub-MIP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x & & \\ \text{s.t.} \quad & Ax \leq b & & \\ & l_j \leq x_j \leq u_j & \text{for all } j \notin \mathcal{I}. & \\ & \lfloor \bar{x}^0_j \rfloor \leq x_j \leq \lceil \bar{x}^0_j \rceil & \text{for all } j \in \mathcal{I}. & \\ & x_j \in \mathbb{Z} & \text{for all } j \in \mathcal{I}. & \end{aligned} \tag{3}$$

In practice, solving (3) may be computationally expensive if only a small percentage of integer variables is assigned tighter bounds. On the contrary, in a considerable amount of cases, RENS tightens too many variable bounds, which leads to the infeasibility of (3) [8]. In the following, we propose to combine RENS with information obtained from Stages 1 and 2 of FP. During the pumping loop, FP generates a sequence $\bar{x}^0, \ldots, \bar{x}^k$ of LP-feasible vectors which can be used as reference vectors for RENS. Let $\bar{X}$ be a subset of $\{\bar{x}^0, \ldots, \bar{x}^k\}$. We propose a modified version of RENS, which we call *multi-reference RENS* (mRENS), in which the variable bounds are tightened based on the values they take in vectors from $\bar{X}$. More precisely, for each $j \in \mathcal{I}$, we set l_j to $\lfloor \min_{\bar{x} \in \bar{X}} \bar{x}_j \rfloor$, and u_j to $\lceil \max_{\bar{x} \in \bar{X}} \bar{x}_j \rceil$. To summarize, mRENS solves the sub-MIP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x & & \\ \text{s.t.} \quad & Ax \leq b & & \\ & l_j \leq x_j \leq u_j & \text{for all } j \notin \mathcal{I}. & \\ & \lfloor \min_{\bar{x} \in \bar{X}} \bar{x}_j \rfloor \leq x_j \leq \lceil \max_{\bar{x} \in \bar{X}} \bar{x}_j \rceil & \text{for all } j \in \mathcal{I}. & \\ & x_j \in \mathbb{Z} & \text{for all } j \in \mathcal{I}. & \end{aligned} \tag{4}$$

Note that if $\bar{x}^0 \in \bar{X}$, then the RENS sub-MIP is itself a sub-MIP of the mRENS sub-MIP.

4 Multiple Reference Vectors in Stages 1 and 2

Next, we investigate how we can use multiple reference vectors in Stages 1 and 2. As already mentioned, the projection step objective considers the distance to the integer vector $\tilde{x}^k$ obtained in the current iteration. From a geometric point of view, the distance function points towards a direction that is likely to reduce the violation of the integrality constraints of the MIP. All its coefficients are in $\{-1, 0, 1\}$ depending on the values of $\tilde{x}^k$, which reduces the set of possible distance functions to 3^n. We enlarge this set by considering a conic combination of multiple distance functions (w.r.t. multiple integer vectors in (2)). Using directions "in-between", may also lead to feasible MIP solutions or rather LP solutions from the projection step that can be easily rounded to such. Further,

they might serve as a "tie-breaker" in dimensions where the current distance function would have a zero coefficient.

Let $\mathcal{L} = \{1, \ldots, k\}$ be the index set of integer vectors obtained during the pumping loop. By $\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L})$, we denote a conic combination of the L_1-distances to the integer vectors $\tilde{x}^i$, $i \in \mathcal{L}$, more precisely,

$$\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L}) := \sum_{i \in \mathcal{L}} \lambda_i \Delta(x, \tilde{x}^i). \tag{5}$$

where $\Lambda_\mathcal{L} := \{\lambda_1, \ldots, \lambda_k\}$, $\lambda_i \geq 0$ for all $i \in \mathcal{L}$, are the conic weights and $\tilde{X}_\mathcal{L} := \{\tilde{x}^1, \ldots, \tilde{x}^k\}$.

We suggest using some variant of (5) as an objective in the projection step of FP. Hence, in the projection step, we have to solve the following LP

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{L}:\, \lambda_i > 0} \lambda_i \left(\sum_{j \in \mathcal{I}: \tilde{x}^i_j = l_j} (x_j - l_j) + \sum_{j \in \mathcal{I}: \tilde{x}^i_j = u_j} (u_j - x_j) + \sum_{j \in \mathcal{I}: l_j < \tilde{x}^i_j < u_j} d^i_j \right) \\
\text{s.t} \quad & x \in P \\
& d^i_j \geq \tilde{x}^i_j - x_j, && i \in \mathcal{L},\, j \in \mathcal{I} : \lambda_i > 0,\, l_j < \tilde{x}^i_j < u_j \\
& d^i_j \geq x_j - \tilde{x}^i_j, && i \in \mathcal{L},\, j \in \mathcal{I} : \lambda_i > 0,\, l_j < \tilde{x}^i_j < u_j \\
& d^i_j \in \mathbb{R}, && i \in \mathcal{L},\, j \in \mathcal{I} : \lambda_i > 0,\, l_j < \tilde{x}^i_j < u_j.
\end{aligned} \tag{6}$$

In (6) each term of the objective function $\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L})$ is linearized separately, as explained in Section 2. A combination with the OFP objective is straightforward. In this case, the objective function of (6) is replaced by

$$\Delta_\alpha(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L}) := (1 - \alpha_k)\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L}) + \frac{\alpha_k \|\Delta\|_2}{\|c\|_2} c^\top x. \tag{7}$$

Here, $\|\Delta\|_2$ is the Euclidean norm of the objective function vector of (6). Note that for the special case where $\lambda_k = 1$ and $\lambda_i = 0$ for all $i \in \mathcal{L} \backslash \{k\}$ the objective (7) is equal to (2).

Another idea is to create a single aggregated reference vector $\tilde{x}^{agg}$, e.g., a convex combination of the integer vectors from previous pumping iterations and replace the distance function in (2) by $\Delta(x, \tilde{x}^{agg})$ Another natural thing to try is to compute a single reference vector as a convex combination of integer vectors However, $\tilde{x}^{agg}$ is not guaranteed to satisfy the integrality constraints, i.e.and therefore will not be considered any further.

The following theorem and corollary motivate why (5) is a reasonable choice to drive the LP solutions towards integrality, and indicate how weights should be chosen.

Theorem 4.1. *Let $\mathcal{L} = \{1, \ldots, k\}$ be the set of indices of $k \in \mathbb{N}$ integer vectors $\tilde{x}^1, \ldots, \tilde{x}^k \in \mathbb{Z}^n$. Further, let $\lambda_i \geq 0$, for all $i \in \mathcal{L}$, and $\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L})$ be defined as in (5). The set of minimizers of $\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L})$ over $\mathbb{R}^n$ always contains an integer vector.*

Proof. See Mexi [10]. □

Remark 4.2. *Since $\Delta(x, \tilde{X}_\mathcal{L}, \Lambda_\mathcal{L})$ is convex, it follows that the set of its minimizers S is also convex. Further, from Theorem 4.1 it follows that S is the Cartesian product of closed intervals with integer boundaries (which might consist of a single integer value, as an important special case).*

Corollary 4.3. *Let $\mathcal{L} = \{1, \ldots, k\}$ be the set of indices of $k \in \mathbb{N}$ integer vectors $\tilde{x}^1, \ldots, \tilde{x}^k \in \mathbb{Z}^n$. Further, let $\lambda_i \geq 0$ for all $i \in \mathcal{L}$, and $\Delta(x, \tilde{X}_{\mathcal{L}}, \Lambda_{\mathcal{L}})$ be defined as in (5). If for any two disjoint sets $U, V \subseteq \mathcal{L}$ with $U \cup V = \mathcal{L}$ it holds that*

$$\sum_{i \in U} \lambda_i - \sum_{i \in V} \lambda_i \neq 0, \tag{8}$$

then the set of minimizers of $\Delta(x, \tilde{X}_{\mathcal{L}}, \Lambda_{\mathcal{L}})$ over $\mathbb{R}^n$ contains only integer vectors.

Corollary 4.3 introduces a condition on the weights $\Lambda_{\mathcal{L}}$, which ensures the integrality of the minimizers of (5). However, this does not imply that any set of weights fulfilling this condition is an appropriate choice for the projection step of FP. For instance, if at any iteration k it holds that $\lambda_1 = \lambda_2 = \cdots = \lambda_{k-1} > \lambda_k$, then (8) is obviously fulfilled. But as k gets larger, the distance function (5) changes less and less, leading to stalling in the FP algorithm. We, therefore, suggest considering only a few reference integer vectors and carefully choosing the weights $\Lambda_{\mathcal{L}}$ to avoid the above behavior. Based on the results of Mexi [10] we consider three reference vectors in any iteration k with geometric weights (0.5, 0.25, 0.125). The first one is $\tilde{x}^k$ and the remaining two are previously obtained integer vectors with the smallest sum of violation over all constraints.

By design, using multiple integer reference vectors in Stages 1 and 2 leads to smaller changes of the distance function in the projection step objective, which can be seen as taking smaller "steps" towards convergence aiming at staying close to good quality feasible solutions. This is expected to come with a slightly increased number of iterations that are traded in for better solution quality. Note that this might not necessarily be desired when running FP with very strict iteration limits.

Considering information from previous iterations in the projection step may increase the probability of cycling, since the distance function in the objective may not change much during the pumping loop. Therefore, it is essential to provide different ways to escape cycles, since an increased number of random perturbations, and especially restarts, may drastically reduce the success rate of FP [6]. For instance, if a cycle of length one is detected, i.e., $\tilde{x}^k = \tilde{x}^{k-1}$, normally, a small number of integers components of $\tilde{x}^k$ are set to a different integer value. One alternative way of breaking this cycle is to adjust the weights $\Lambda_{\mathcal{L}}$ and repeat the iteration $k-1$. Intuitively, since $\tilde{x}^{k-1}$ is revisited, we reduce the value of λ_{k-1}, i.e., reduce the contribution of $\Delta(x, \tilde{x}^{k-1})$ in (5). In case that a cycle of length greater than one is detected, i.e., $\tilde{x}^k = \tilde{x}^l$ for some $l \in \{1, \ldots, k-2\}$, the restart mechanism can be avoided in a similar way. For instance, if $l \in \mathcal{L}_{\text{active}}$, we decided to use the original projection step of FP for one iteration by setting $\lambda_k = 1$, and $\lambda_i = 0, i \in \mathcal{L} \backslash \{k\}$ to ignore all information from previous iterations. Lastly, if the cycle persists, or a new cycle is detected, we apply the usual restart procedure of FP.

5 Alternative Objective Scaling

After experimenting with the problem instance `misc07`, Pradignac et al. [11] observed that for this instance the fixed scaling terms $\|\Delta\|_2$ and $\|c\|_2$ do not provide a balanced contribution of $\Delta(x, \tilde{x}^k)$ and $c^\top x$ in (2). Therefore, they

proposed to replace $\|c\|_2$ by $\left|c^\top \bar{x}^{k-1}\right|$, where $\bar{x}^{k-1}$ is the LP-feasible vector obtained in the projection step of the previous iteration. This modification resulted in a more balanced contribution of the two objective terms for this specific problem instance. However, no comparison over a large test set was presented.

Inspired by this idea, we propose to replace the scaling term $\|\Delta\|_2$ by the dynamically changing term $\left|\Delta(\bar{x}^{k-1}, \tilde{x}^k)\right|$ in addition to the scaling suggested above. In the spirit of this work, one can think of these two modifications as considering information from previous iterations in the projection step.

Recall that FP aims at minimizing the distance of the vectors $\bar{x}^k$ and $\tilde{x}^k$, and reach convergence. Thus, it is expected that the components of $\bar{x}^k$ and $\tilde{x}^k$ do not differ much from the corresponding components of $\bar{x}^{k-1}$ and $\tilde{x}^{k-1}$. Consequently, in the projection step we also expect that $\Delta(x, \tilde{x}^k)$ and $c^\top \bar{x}^k$ will acquire values close to $\Delta(\bar{x}^{k-1}, \tilde{x}^k)$ and $c^\top \bar{x}^{k-1}$, respectively. Therefore, the suggested scaling terms should provide a more balanced contribution of the two objective terms. To summarize, we propose the modified projection step objective given by

$$\Delta_{\alpha_k}(x, \tilde{x}^k) := (1 - \alpha_k)\Delta(x, \tilde{x}^k) + \frac{\alpha_k \left|\Delta(\bar{x}^{k-1}, \tilde{x}^k)\right|}{|c^\top \bar{x}^{k-1}|} c^\top x, \tag{9}$$

where $\alpha_k \in [0, 1]$. Note that a small value for $\left|\Delta(\bar{x}^{k-1}, \tilde{x}^k)\right|$ might also be an indicator of convergence in subsequent iterations. Whenever $\left|\Delta(\bar{x}^{k-1}, \tilde{x}^k)\right|$ decreases, the contribution of $c^\top x$ in the objective also decreases, which turns the priority of the projection step more towards feasibility and less towards solution quality. Therefore, dynamically adjusting the two scaling terms may also accelerate the convergence to a feasible solution.

6 Setup

As a baseline for our C++ implementation we use the FP code from [7], that incorporates the WalkSAT perturbation procedure described in Dey et al. [12]. In the implementation, FP communicates with LP/MIP solvers through an abstract interface, which is implemented for two state-of-the-art commercial solvers, namely, the FICO Xpress Optimizer [13] and IBM ILOG CPLEX [14]. In addition, we provide a solver interface for the non-commercial solver SCIP using SoPlex [15] for solving LPs. To the best of our knowledge, this is the first state-of-the-art OFP2 implementation that only uses open-source software. Our final version of the code is freely available and can be downloaded from `https://github.com/GioniMexi/feaspump3`.

Similar to [7] we compare variants of FP in an *embedded* and a *standalone* scenario. In the first case, we evaluate FP as a primal heuristic embedded in a general-purpose MIP solver. Typically solvers do not spend a lot of computing time on a single heuristic. Therefore, in this scenario, we set a limit of 20 iterations in Stages 1 and 2. In the second scenario, we evaluate the performance of each FP variant as a standalone primal heuristic. Here the total number of iterations in Stage 1 is 10000 iterations, and in Stage 2, it is 2000 iterations. A detailed description of the parameter settings that we use for our standalone experiment can be found in [5, 6]. The chosen iteration limits align with those in earlier implementations, in particular the FP 2.0. In preliminary experiments,

we also tried slightly different limits (e.g., 50 or 100 iterations for Stages 1 and 2) and found that this does not change the overall picture by much.

In both scenarios, embedded and standalone, when we evaluate the performance of mRENS and Stage 3, we limit the number of explored MIP nodes to 500. This coincides with typical node limits applied for sub-MIP solving heuristics in SCIP, see, e.g., [8, 16, 17], and is in marked difference to previous implementations of FP heuristics which either avoided a sub-MIP stage altogether (when used in an embedded setting) or did not apply any resource limitations beyond a time limit (when used in a standalone setting).

Our test set consists of the official benchmark set `MIPLIB` 2017 [18] and all instances used in [7] that are not part of `MIPLIB` 2017. We excluded the instances `decomp2`, `ex9`, and `ex10`, since they were solved to optimality during presolve, and `supportcase19`, since solving its LP relaxation exceeded one hour of computation. In total, our main test set consists of 283 instances, for a detailed list of all instances see our online supplement `https://github.com/GioniMexi/online-supplement-feaspump3`.

All results were obtained on a cluster of Intel Xeon E5-2690 @ 2.6 GHz machines with 128 GB of RAM, using a time limit of 3600 seconds, and exclusive runs for each job. In all experiments, we use the FICO Xpress Optimizer as LP and MIP solver. To mitigate the impact of performance variability, each FP variant is tested on the complete test set with three different seeds. Therefore, we decided to use average results over all three seeds to compare different FP variants. For details about performance variability in mixed-integer programming, we refer to Lodi and Tramontani [19].

7 Computational Results

For our comparisons, we use the so-called shifted geometric mean [20], with a shift of 1. All FP variants are tested over the whole test set with the same three random seeds. In our tables, we always report the number of instances for which an FP variant finds a solution with all, some, and none of the seeds. Further, we compute the primal gap (gap(%)), running time (time(s)), and number of iterations (iter) as the shifted geometric mean over all instances solved by at least one of the FP variants that we compare.

In all tables, we highlight values in blue and bold that are at least 10% better than the values obtained by the reference runs, which are always the first row of each table. Values that are at least 10% worse than the reference are highlighted in red and in italics.

Over all 849 instance-seed combinations, we count the number of wins, i.e., the number of instances an FP variant provides at least a 10% relative improvement, for the primal gap, running time, and the number of iterations. As before, if the number of wins exceeds the number of losses by more than 10%, the number of wins is highlighted in blue and bold. If it is the other way around, the number of losses is highlighted in red and italics.

In Sections 7.1 to 7.3 we examine the performance of different FP variants in an embedded setting and in Section 7.4 in a standalone setting. For the complete results, we refer to our online supplement `https://github.com/GioniMexi/online-supplement-feaspump3`. Note that for all different variants, we use OFP2 as a baseline and hence FP in setting abbreviations will always mean

that a variant of OFP2 was used. We present our results in the order of the paper outline.

7.1 Applying mRENS

First, we illustrate the benefits of using (m)RENS before Stage 3. To this end, we compare FP with activated Stage 3 (FP-(s3)) to the variant which applies either RENS or mRENS, called FP-(RENS,s3) and FP-(mRENS,s3), respectively. In this setting, if (m)RENS finds a feasible solution, we terminate. Otherwise, we use the original Stage 3 for the remaining available time. Our experiment is conducted over the 153 instances (out of 283) where FP fails to find a feasible solution within the limit of 20 iterations.

The results in Table 1 show that both new variants find feasible solutions for slightly more instances than FP-(s3). In addition, FP-(mRENS,s3) significantly outperforms FP-(s3) w.r.t. solution quality. More precisely, it leads to a 41.8% gap decrease over instances solved by at least one FP variant, however with a slight increase of computational time since in many cases two sub-MIPs are solved. The geometric mean of the time spent in the sub-MIP stages of FP is shown in the parentheses in the last column of the table. In particular, RENS and mRENS are able to find a solution with all three seeds for 42% and 54% of the instances and hence skip Stage 3. Both RENS and mRENS are significantly faster than the original Stage 3. Namely, in terms of geometric means RENS requires 0.24 seconds, and mRENS 0.67 seconds.

As already mentioned, many implementations of FP in MIP solvers skip the computationally demanding Stage 3. We thus suggest that mRENS is a good alternative for Stage 3 in embedded implementations of FP, since it require 62.8% less computational time. However, similar to the RENS heuristic, there is no guarantee that the defined sub-MIP is going to be feasible. Fortunately, in most cases state-of-the-art MIP solvers are able to detect infeasibility in a short amount of time, which fits the "fail fast" design concept of primal heuristics [1]. It is worth pointing out that the time spent in the sub-MIP heuristics (s3)/(RENS,s3)/(mRENS,s3) was less than the time spent in Stages 1 and 2 on average in all three cases, supporting the claim that we can afford Stage 3, with appropriate limits, even in the embedded setting.

Table 1: Comparison of using RENS and mRENS before Stage 3 over the 153 instances for which no feasible solution is found in Stage 1 and 2 after 20 iterations. Geometric means are highlighted blue-bold if they indicate a 10% improvement or red-italic if they indicate a 10% deterioration. The numbers in parentheses in the last column show the combined time spent in (m)RENS and Stage 3.

	solved (#seeds)			#wins/#losses vs FP		geom. mean	
	all	some	none	gap(%)	time(s)	gap(%)	time(s)
FP-(s3)	127	16	10	-	-	35.88	5.54(1.80)
FP-(RENS,s3)	129	14	10	**145**/21	*57*/86	**22.58**	5.71(2.00)
FP-(mRENS,s3)	131	13	9	**198**/25	*23*/63	**20.87**	5.78(1.83)

7.2 Applying the New Scaling

Next, we compare FP with configurations that use the alternative scaling terms introduced in Section 5. Here, we deactivate Stage 3 to better compare the changes in Stage 1 and 2. Table 2 summarizes the results of three runs: (1) FP, (2) FP using the alternative scaling for $\|c\|_2$ (FP-(c)), and (3) FP using the alternative scalings for both $\|c\|_2$ and $\|\Delta\|_2$ (FP-(c,Δ)).

Both FP-(c) and FP-(c,Δ) lead to a significant performance improvement. Especially FP-(c,Δ) leads to a 33.6% gap decrease over instances solved by at least one FP variant and requires 15.8% fewer iterations than FP to find a feasible solution.

A further impressive result is that FP-(c,Δ) is able to find a feasible solution for 33.3% more instances than FP in less than 20 iterations. Our results suggest that replacing FP with FP-(c,Δ) in MIP solvers could be beneficial, especially for solvers that use FP only for a few iterations.

Table 2: Comparison of FP and variants that use the alternative objective scaling in an embedded setting. Geometric means are highlighted blue-bold if they indicate a 10% improvement.

	solved (#seeds)			#wins/#losses vs FP			geom. mean		
configuration	all	some	none	gap(%)	time(s)	iter	gap(%)	time(s)	iter
FP	99	31	153	-	-	-	29.16	3.0	11.0
FP-(c)	104	36	143	**110**/91	**211**/144	**123**/85	**24.31**	3.02	10.67
FP-(c,Δ)	132	30	121	**174**/81	**268**/143	**222**/76	**19.37**	2.89	**9.26**

7.3 Final Embedded Setting

Since in the embedded setting both mRENS and the alternative objective scaling provide a significant performance improvement, we incorporate both in FP and make a final comparison. In Table 3 we compare FP-(s3) with the combination of FP-(c,Δ) and FP-(mRENS,s3). Here FP-(c,Δ)-(mRENS,s3) undoubtedly outperforms FP-(s3). It provides a 29.3% gap decrease over instances solved by at least one of the two algorithms, and a 4.0% decrease in running time requires 16.0% fewer iterations.

Table 3: Comparison of FP to FP-(c,Δ)-(mRENS,s3). Geometric means are highlighted blue-bold if they indicate a 10% improvement.

	solved (#seeds)			#wins/#losses vs FP			geom. mean		
configuration	all	some	none	gap	time(s)	iter	gap	time(s)	iter
FP-(s3)	256	17	10	-	-	-	25.56	4.26	11.0
FP-(c,Δ)-(mRENS,s3)	259	11	13	**327**/133	**340**/194	**222**/72	**18.06**	4.09	**9.24**

7.4 Standalone Results

As it has been shown in the thesis of Mexi [10], the new scaling terms and the use of multiple reference vectors in Stages 1 and 2 enhance the performance of a standalone FP implementation in a complementary way. In particular, the new scaling terms mostly lead to an improvement in terms of running time and iterations, whereas using multiple reference vectors in the projection step improves

the solution quality. Based on this, it seems natural to compare combinations of different FP variants in a standalone setting.

In the first part of Table 4 we compare FP to FP-(c,Δ)-(s3), FP-(mrv)-(s3) which uses three integer reference vectors in the projection step, and FP-(mrv)-(c,Δ)-(s3) which uses both the alternative scaling and multiple reference vectors. Here, even though FP-(c,Δ)-(s3) is faster and requires significantly fewer iterations than FP, the solution quality is slightly worse, namely, the gap is 3.1% worse than the gap obtained by FP. On the other hand, using multiple integer reference vectors (FP-(mrv)-(s3)) leads to a 5.4% gap improvement but to a worsening of the running time and iterations. Combining both approaches (FP-(mrv)-(c,Δ)-(s3)) we obtain more wins w.r.t. the gap, and a decrease in the total number of iterations. By enhancing the previous versions of FP by mRENS the gap improves further, especially for FP-(mrv)-(mRENS,s3) where the gap improves by 10.8%.

Table 4: Comparison of FP to standalone variants. Geometric means are highlighted blue-bold if they indicate a 10% improvement or red-italic if they indicate a deterioration.

	solved (#seeds)			#wins/#losses vs FP			geom. mean		
	all	some	none	gap	time(s)	iter	gap	time(s)	iter
FP-(s3)	268	6	9	-	-	-	21.27	6.41	28.93
FP-(c,Δ)-(s3)	266	4	13	*171*/211	**310**/185	**325**/177	21.93	6.15	**22.88**
FP-(mrv)-(s3)	266	7	10	**205**/132	*147*/270	*118*/307	20.12	6.95	*35.02*
FP-(mrv)-(c,Δ)-(s3)	266	6	11	**213**/204	**309**/254	**294**/247	21.28	6.32	26.34
FP-(c,Δ)-(mRENS,s3)	267	7	9	210/192	**301**/210	**325**/177	20.17	6.21	**22.88**
FP-(mrv)-(mRENS,s3)	267	6	10	**232**/123	*131*/276	*118*/307	**18.98**	7.0	*35.02*
FP-(mrv)-(c,Δ)-(mRENS,s3)	268	4	11	**223**/202	**278**/231	**294**/247	20.94	6.19	26.34

To conclude, our results indicate that in the standalone case using the new scaling, three integer reference vectors in Stage 1 and 2, and applying mRENS before Stage 3, leads to both solution quality and running time improvement.

8 Conclusion

In this paper, we presented a new version of the Objective Feasibility Pump 2 heuristic that makes use of multiple reference vectors, in all three stages of the algorithm. In particular, we introduced a modified version of RENS, which we call mRENS that uses LP-feasible reference vectors obtained in the main loop of FP to tighten the variable bounds before entering the computationally expensive Stage 3. We extended the idea of using multiple reference vectors to Stage 1 and 2. This is achieved by using a conic combination of distance functions in the projection step objective. Moreover, we introduced alternative objective scaling terms to balance the contributions of the distance functions and the original MIP objective $c^\top x$. In addition, we provide a solver interface for the non-commercial solver SCIP using SoPlex [15] for solving LPs. To the best of our knowledge, this is the first state-of-the-art FP implementation that only uses open-source software. Our final version of the code is freely available and can be downloaded from `https://github.com/GioniMexi/feaspump3`.

Our computational experiments demonstrate that the new method can improve the performance on general MIP test sets. In detail, our modifications provide in an embedded setting a 29.3% solution quality improvement and 4.0% running time improvement, needing 16.0% fewer iterations over a large test set of MIP instances. In addition, the success rate of the method within the first few iterations increases considerably. In a standalone setting, we observed also a moderate performance improvement for the gap, running time, and iterations.

References

[1] Timo Berthold. *Heuristic algorithms in global MINLP solvers*. PhD thesis, Technische Universität Berlin, 2014. `http://www.zib.de/berthold/Berthold2014.pdf`.

[2] Timo Berthold. Primal heuristics for mixed integer programs. Master's thesis, Technische Universität Berlin, 2006. `http://nbn-resolving.de/urn:nbn:de:0297-zib-10293`.

[3] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[4] Timo Berthold, Andrea Lodi, and Domenico Salvagnin. Ten years of feasibility pump, and counting. *EURO Journal on Computational Optimization*, 7(1):1–14, 2019.

[5] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007.

[6] Tobias Achterberg and Timo Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.

[7] Matteo Fischetti and Domenico Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computation*, 1(2-3):201–222, 2009.

[8] Timo Berthold. RENS. *Mathematical Programming Computation*, 6(1):33–54, 2013.

[9] Timo Berthold. RENS – Relaxation Enforced Neighborhood Search. ZIB-Report 07-28, Zuse Institute Berlin, 2007.

[10] Gioni Mexi. New ideas for the feasibility pump: Using multiple reference vectors. Master's thesis, 2021.

[11] Nicolas Pradignac, Maliheh Aramon, and Helmut G Katzgraber. A feasibility pump algorithm embedded in an annealing framework, 2019. preprint arXiv:1906.06434.

[12] Santanu S Dey, Andres Iroume, Marco Molinaro, and Domenico Salvagnin. Improving the randomization step in feasibility pump. *SIAM Journal on Optimization*, 28(1):355–378, 2018.

[13] FICO. Xpress Optimizer v8.11 reference manual. `www.fico.com/fico-xpress-optimization/docs/latest/overview.html`, 2021.

[14] IBM. ILOG CPLEX v12.10 reference manual. `www.ibm.com/support/knowledgecenter/SSSA5P_12.10.0/COS_KC_home.html`, 2020.

[15] Roland Wunderling. *Paralleler und objektorientierter Simplex*. PhD thesis, Technische Universität Berlin, 1996. `http://nbn-resolving.de/urn:nbn:de:0297-zib-5386`.

[16] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005.

[17] Gregor Hendel. Adaptive large neighborhood search for mixed integer programming. *Mathematical Programming Computation*, pages 1–37, 2021.

[18] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, pages 1–48, 2021.

[19] Andrea Lodi and Andrea Tramontani. Performance variability in mixed-integer programming. In *Theory Driven by Influential Applications*, pages 1–12. INFORMS, 2013.

[20] Tobias Achterberg. *Constraint integer programming*. PhD thesis, Technische Universität Berlin, 2007.