Improving the Randomization Step in Feasibility Pump

Santanu S. Dey*,1, Andres Iroume†1, Marco Molinaro‡2, and Domenico Salvagnin§3

1School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, United States
2Computer Science Department, PUC-Rio, Brazil
3IBM Italy and DEI, University of Padova, Italy

June 22, 2017

Abstract

Feasibility pump is a successful primal heuristic for mixed-integer linear programs. The algorithm consists of three main components: rounding fractional solution to a mixed-integer one, projection of infeasible solutions to the Linear Programming relaxation, and a randomization step used when the algorithm stalls. While many generalizations and improvements to the original Feasibility Pump have been proposed, they mainly focus on the rounding and projection steps.

We start a more in-depth study of the randomization step in Feasibility Pump. For that, we propose a new randomization step based on the WalkSAT algorithm for solving instances of the Boolean Satisfiability Problem. First, we provide theoretical analyses for instances with disjoint equality constraints that show the potential of this randomization step; to the best of our knowledge, this is the first time any theoretical analysis of running-time of Feasibility Pump or its variants has been conducted, even for a special class of instances. Moreover, we propose a practical version of new randomization step, and incorporate it into a state-of-the-art Feasibility Pump code. Our experiments suggests that this simple-to-implement modification consistently dominates the standard randomization previously used.

1 Introduction

Primal heuristics are used within mixed-integer linear programming (MILP) solvers for finding good integer feasible solutions quickly [FL11]. Feasibility pump (FP) is a very successful primal heuristic for mixed-binary LPs that was introduced in [FGL05]. At its core, Feasibility Pump is an alternating projection method, as described below.

Algorithm 1 Feasibility Pump (Naïve version)

1: \textbf{Input}: mixed-binary LP (with binary variables }x\text{ and continuous variables }y\text{)
2: Solve the linear programming relaxation, and let } (\bar{x}, \bar{y}) \text{ be an optimal solution}
3: \textbf{while } \bar{x} \text{ is not integral \textbf{do}}
4: \hspace{1em} (Round) Round each coordinate of } \bar{x} \text{ to the closest integer, call the obtained vector } \tilde{x}
5: \hspace{1em} (Project) Let } (\tilde{x}, \bar{y}) \text{ be the point in the LP relaxation that minimizes } \sum_{i} |x_{i} - \tilde{x}_{i}|
6: \textbf{end while}
7: \textbf{Return} } (\tilde{x}, \bar{y})

The scheme presented above may stall, since the same infeasible integer point may be visited in Step 4 at different iterations. Whenever this happens, the paper [FGL05] recommends a randomization step,
that after Step 4 flips the value of some of the binary variables as follows: Defining the fractionality
of variable \( x_i \) as \( |\bar{x}_i - \tilde{x}_i| \) and let \( N \) be the number of variables with positive fractionality, randomly
generate a positive integer \( T \) and flip \( \min(T, N) \) variables with largest fractionality.

Together with a few other tweaks, this surprisingly simple method works very well. On MIPLIB 2003
instances, FP finds feasible solutions for 96.3% of the instances in reasonable time [FGL05].

Due to its success, many improvements and generalizations of FP, both for MILPs and mixed integer
non-linear programs (MINLPs), have been studied [AB07, BFL07, BCLM09, FS09, SLR13, DFLL10,
BEET12, DFLL12, BEE14]. However, the focus of these improvements has been on the projection and
rounding steps or generalizations for MINLPs; to the best of our knowledge, they use essentially the
same randomization step as proposed in the original algorithm [FGL05] (and its generalization to the
general integer MILP case of [BFL07]). We note that some approached avoid the randomization step
altogether, see [BCLM09] for MINLPs and [GMSS17] MIPs.

Moreover, even though FP is so successful and so many variants have been proposed, there is very
limited theoretical analysis of its properties [BEET12]. In particular, to the best of our knowledge there
is no known bounds on expected running-time of FP.

2 Our contributions

In this paper, we start a more in-depth study of the randomization step in Feasibility Pump. For that,
we propose a new randomization step \( \text{RandWalkSAT}_\ell \) and provide both theoretical analysis as well
as computational experiments in a state-of-the-art Feasibility Pump code that show the potential of this
method.

**Theoretical justification of RandWalkSAT\(_\ell\).** The new randomization step \( \text{RandWalkSAT}_\ell \) is
inspired by the classical algorithm WalkSAT [Sch99] for solving instances of the Boolean Satisfiability
Problem (SAT) (see also [Pap91, MJPL92]). The key idea of \( \text{RandWalkSAT}_\ell \) is that whenever
Feasibility Pump stalls, namely an infeasible mixed-binary solution is revisited, it should flip a binary
variable that participates in an infeasible constraint. More precisely, \( \text{RandWalkSAT}_\ell \) constructs a
minimal (projected) infeasibility certificate for this solution and randomly picks \( \ell \) binary variable in it to
be flipped (see Section 3 for exact definitions).

While the vague intuition that such randomization is trying to “fix” the infeasible constraint is clear,
we go further and provide theoretical analyses that formally justify this and highlight more subtle
advantageous properties of \( \text{RandWalkSAT}_\ell \).

First, we analyze what happens if we simply repeatedly use only the new proposed randomization
step \( \text{RandWalkSAT}_\ell \), which gives a simple primal heuristic that we denote by \( \text{mbWalkSAT} \) (“MB"
indicates it is an extension of WalkSAT for mixed-binary problems). Not only we show that \( \text{mbWalkSAT} \)
is guaranteed to find a solution if one exists, but its behavior is related to the \((almost)\) decomposability
and sparsity of the instance. To make this precise, consider a decomposable mixed-binary set with \( k \)
blocks:

\[
P^I = P_1^I \times \ldots \times P_k^I, \text{ where for all } i \in [k] \text{ we have } \\
P_i^I = P_i \cap \{(0,1)^{n_i} \times \mathbb{R}^{d_i}\}, P_i = \{(x^i, y^i) \in [0,1]^{n_i} \times \mathbb{R}^{d_i} : A^i x^i + B^i y^i \leq c^i\}. \tag{1}
\]

Let \( P = P_1 \times \ldots \times P_k \) denote the LP relaxation of \( P^I \).

Note that since we allow \( k = 1 \), this also captures a general mixed-binary set. We then have the following
running-time bound for the primal heuristic \( \text{mbWalkSAT} \).

**Theorem 2.1.** Consider a feasible decomposable mixed-binary set as in equation (1). Let \( s_i \) be such
that each constraint in \( P^I \) has at most \( s_i \) binary variables, and define \( \gamma_i := \min\{s_i \cdot (d_i + 1), n_i\} \).
Then with probability at least \( 1 - \delta \), \( \text{mbWalkSAT} \) with parameter \( \ell = 1 \) returns a feasible solution
within \( \ln(k/\delta) \sum_i n_i 2^{n_i \log \gamma_i} \) iterations. In particular, this bound is at most \( nk 2^{n \log \bar{n}} \cdot \ln(k/\delta) \), where \( \bar{n} = \max_i n_i \).

There are a few interesting features of this bound that indicates good properties of the proposed
randomization step, apart from the fact that it is already able to find feasible solutions by itself. Suppose
we have a decomposable instance where each of the \( k \) blocks has \( n/k \) variables. If we perform total enumeration without the knowledge of decomposability, in the worst case we might end up trying all \( 2^n \) possible solutions. In contrast, as we see in Theorem 2.1, even without the knowledge of decomposability \( \text{mbWalkSAT} \) takes at most approximately \( n2^{n/k} \) iterations (which for larger number of blocks \( k \) is significantly better than \( 2^n \)). The fact the algorithm does not explicitly use the knowledge of the decomposability of the instances gives some indication that the proposed randomization could still exhibit good behavior on the almost decomposable instances often found in practice (see discussion in [DMW16]).

Finally, notice that the running-time of the algorithm depends on the sparsity \( s_i \) of the blocks, giving slightly better running times on sparser problems.

**RandWalkSAT, in conjunction with FP.** Next, we analyze \( \text{RandWalkSAT} \) in the context of Feasibility Pump by adding it as a randomization step to the Naïve Feasibility Pump algorithm (Algorithm 1); we call the resulting algorithm WFP. This now requires understanding the complicated interplay of the randomization, rounding, and projection steps: While in practice rounding and projection greatly help finding feasible solutions, their worst-case behavior is difficult to analyze and in fact they could take the iterates far away from feasible solutions. Although the general case is elusive at this point, we are nonetheless able to analyze the running time of WFP for decomposable 1-row mixed-binary programs.

**Definition 2.2.** A decomposable 1-row set is a decomposable set as in (1) where each block \( P_i \) has a single equality:

\[
P_i = \{ (x^i, y^i) \in [0, 1]^{n_i} \times \mathbb{R}_+^d : a^i x^i + b^i y^i = c_i \}.
\]

In particular, this class of instances includes subset-sum instances (i.e. \( \{ x \in \{0, 1\}^n : ax = c \} \) with non-negative \( a, c \)) and knapsacks in standard form (i.e. \( \{(x, y) \in \{0, 1\}^n \times \mathbb{R}_+ : ax + by = c \} \) with \( a, c \) non-negative and \( b \in \{-1, 1\} \)). While this may still seem like a simple class of problems, on these instances Feasibility Pump with the original randomization step from [FGL05] (without the restart operation that flips all variables with non-zero probability [BFL07]) may not even converge, as illustrated next.

**Remark 2.3.** Consider the feasible subset-sum problem

\[
\begin{align*}
\max & \quad x_2 \\
\text{s.t.} & \quad 3x_1 + x_2 = 3 \\
& \quad x_1, x_2 \in \{0, 1\}.
\end{align*}
\]

Consider the execution of the original Feasibility Pump algorithm (without restarts). The starting point is an optimal LP solution; without loss of generality, suppose it is the solution \((2/3, 1)\). This solution is then rounded to the point \((1, 1)\), which is infeasible. This point is then \( \ell_1 \)-projected to the LP, giving back the point \((2/3, 1)\), which is then rounded again to \((1, 1)\). At this point the algorithm has stalled and applies the randomization step. Since only variable \( x_1 \) has strictly positive fractionality \( |2/3 - 1| = 1/3 \), only the first coordinate of \((1, 1)\) is a candidate to be flipped. So suppose this coordinate is flipped. The infeasible point \((0, 1)\) obtained is then \( \ell_1 \)-projected to the LP, giving again the point \((2/3, 1)\). This sequence of iterates repeats indefinitely and the algorithm does not find the feasible solution \((1, 0)\).

The issue in this example is that the original randomization step never flips a variable with zero fractionality. Moreover, in Section A of the appendix we show that even if such flips are considered, there is a more complicated subset-sum instance where the algorithm stalls.

On the other hand, we show that algorithm WFP with the proposed randomization step always finds a feasible solution of feasible decomposable 1-row instances, and, moreover, its running-time again depends on the sparsity and the decomposability of the instance.

**Theorem 2.4.** Consider a feasible decomposable 1-row set. Then with probability at least \( 1 - \delta \), WFP with \( \ell = 2 \) applied to this set returns a feasible solution within

\[
T = \lceil \ln(k/\delta) \rceil \sum_i n_i (n_i + 1) \cdot 2^{2n_i \log n_i} \leq \lceil \ln(k/\delta) \rceil \cdot k(n + 1)^2 \cdot 2^{2n \log n}
\]

iterations, where \( \bar{n} = \max_i n_i \).
This result is proved in Section 4.1. To the best of our knowledge this is the first theoretical analysis of the running-time of a variant of Feasibility Pump algorithm, even for a special class of instances. As in the case of repeatedly using just RandWalkSAT, the algorithm WFP essentially works independently on each of the blocks (inequalities) of the problem, and has reduced running time on sparser instances.

The high-level idea of the proof Theorem 2.4 is the following. First, we notice that the projection, rounding, and perturbation operators used in the algorithm act independently on each of the blocks of a decomposable instance; this allows us to focus on analyzing the algorithm on just one of the blocks, namely a 1-row problem. To perform this analysis, we: 1) Show that these instances there can only be sequences of at most $n + 1$ consecutive ‘projection plus rounding’ operations (Corollary 4.8), after which the algorithm either returns or stalls; 2) Show that a round of ‘randomization step plus projection plus rounding’ has a non-zero probability of generating an iterate closer to a coordinate-wise maximal feasible solution (Lemma 4.12), so the algorithm has some chance of ‘un-stalling’ to a point closer to a feasible solution.

Computational experiments. While the analyses above give insights on the usefulness of using RandWalkSAT, in the randomization step of FP, in order to attest its practical value it is important to understand how it interacts with complex engineering components present in current Feasibility Pump codes. To this end, we considered the state-of-the-art code of [FS09] and modified its randomization step based on RandWalkSAT. While the full details of the experiments are presented in Section 5, we summarize some of the main findings here.

We conducted experiments on MIPLIP 2010 [KAA+11] instances and on randomly generated two-stage stochastic models. In the first testbed there was a small but consistent improvement in both running-time and number of iterations. More importantly, the success rate of the heuristic improved consistently. In the second testbed, the new algorithm performs even better, according to all measures. It is somewhat surprising that our small modification of the randomization step could provide noticeable improvements over the code in [FS09], specially considering that it already includes several improvements over the original Feasibility Pump (e.g. constraint propagation). In addition, the proposed modification is generic and could be easily incorporated in essentially any Feasibility Pump code. Moreover, for virtually all the seeds and instances tested the modified algorithm performed better than the original version in [FS09]; this indicates that, in practice, the modified randomization step dominates the previous one.

The rest of the paper is organized as follows: Section 3 we discuss and present our analysis of the proposed randomization scheme RandWalkSAT, Section 4 presents the analysis of the new randomization scheme RandWalkSAT in conjunction with feasibility pump, and Section 5 describes details of our empirical experiments.

Notation. We use $\mathbb{R}_+$ to denote the non-negative reals, and $[k] := \{1, 2, \ldots, k\}$. For a vector $v \in \mathbb{R}^n$, we use $\text{supp}(v) \subseteq [n]$ to denote its support, namely the set of coordinates $i$ where $v_i \neq 0$. We also use $\|v\|_0 = |\text{supp}(v)|$, and $\|v\|_1 = \sum |v_i|$ to denote the $\ell_1$ norm. Finally, we use $e^i \in \mathbb{R}^n$ to denote the $i$th canonical basis vector.

3 New randomization step RandWalkSAT

3.1 Description of the randomization step

We start by describing the WalkSAT algorithm [Sch99], that serves as the inspiration for the proposed randomization step RandWalkSAT, in the context of pure-binary linear programs. The vanilla version of WalkSAT starts with a random point $\bar{x} \in \{0, 1\}^n$; if this point is feasible, the algorithm returns it, and otherwise selects any constraint violated by it. The algorithm then select a random index $i$ from the support of the selected constraint and flips the value of the entry $\bar{x}_i$ of the solution. This process is repeated until a feasible solution is obtained. It is known that this simple algorithm finds a feasible solution in expected time at most $2^n$ (see [MU05] for a proof for 3-SAT instances), and Schöning [Sch99] showed that if the algorithm is restarted at every $3n$ iterations, a feasible solution is found in expected time at most a polynomial factor from $(2(1 - \frac{1}{2^s}))^n$, where $s$ is the largest support size of the constraints.

Based on this WalkSAT algorithm, to obtain a randomization step for mixed-binary problems we are going to work on the projection onto the binary variables, so instead of looking for violated constraints
we look for a certificate of infeasibility in the space of binary variables. Importantly, we use a minimal certificate, which makes sure that for decomposable instances the certificate does not “mix” the different blocks of the problem.

Now we proceed with a formal description of the proposed randomization step \textsc{RandWalkSAT}_\ell. Consider a mixed-binary set

\[ P^I = P \cap (\{0,1\}^n \times \mathbb{R}^d), \text{ where } P = \{(x,y) \in [0,1]^n \times \mathbb{R}^d : Ax + By \leq c\}. \tag{2} \]

We use \text{proj}_{\text{bin}} P to denote the projection of \(P\) onto the binary variables \(x\).

**Definition 3.1 (Projected certificates).** Given a mixed-binary set \(P^I\) as in (2) and a point \((\bar{x}, \bar{y})\) \(\in\) \(\{0,1\}^n \times \mathbb{R}^d\) such that \(\bar{x} \notin \text{proj}_{\text{bin}} P\), a projected certificate for \(\bar{x}\) is an inequality \(\lambda Ax + \lambda By \leq \lambda b\) with \(\lambda \in \mathbb{R}^m_+\) such that: (i) \(\bar{x}\) does not satisfy this inequality; (ii) \(\lambda B = 0\). A minimal projected certificate is one where the support of the vector \(\lambda\) is minimal (i.e. the certificate uses a minimal set of the original inequalities).

Standard Fourier-Motzkin theory guarantees us that projected certificates always exist, and furthermore Caratheodory’s theorem [Sch86] guarantees that minimal projected certificates use at most \(d + 1\) inequalities. Together these give the following lemma.

**Lemma 3.2.** Consider a mixed-binary set \(P^I\) as in (2) and a point \((\bar{x}, \bar{y})\) \(\in\) \(\{0,1\}^n \times \mathbb{R}^d\) such that \(\bar{x} \notin \text{proj}_{\text{bin}} P\). There exists a vector \(\lambda \in \mathbb{R}^m_+\) with support of size at most \(d + 1\) such that \(\lambda Ax + \lambda By \leq \lambda b\) is a minimal projected certificate for \(\bar{x}\). Moreover, this minimal projected certificate can be obtained in polynomial-time (by solving a suitable LP).

Now we can formally define the randomization step \textsc{RandWalkSAT}_\ell (notice that the condition \(\lambda B = 0\) guarantees that a projected certificate has the form \(\pi x \leq \pi_0\)).

**Algorithm 2** \textsc{RandWalkSAT}_\ell(\(\bar{x}\))

1: //Assumes that \(\bar{x}\) does not belong to \text{proj}_{\text{bin}} P
2: Let \(\pi x \leq \pi_0\) be a minimal projected certificate for \(\bar{x}\)
3: Sample \(\ell\) indices from the support \text{supp}(\pi) uniformly and independently, let \(I\) be the set of indices obtained
4: (Flip coordinates) For all \(i \in I\), set \(\bar{x}_i \leftarrow 1 - \bar{x}_i\)

Note that in the pure-binary case and \(\ell = 1\), this is reduces to the main step executed during \textsc{WalkSAT}. We remark that the flexibility of introducing the parameter \(\ell\) will be needed in Section 4.

### 3.2 Analyzing the behavior of \textsc{RandWalkSAT}_\ell

In this section we consider the behavior of the algorithm \textsc{mbWalkSAT} that tries to find a feasible mixed-binary solution by just repeatedly applying the randomization step \textsc{RandWalkSAT}_\ell.

**Algorithm 3** \textsc{mbWalkSAT}

1: **input parameter:** Integer \(\ell \geq 1\)
2: (Starting solution) Consider any mixed-binary point \((\bar{x}, \bar{y}) \in \{0,1\}^n \times \mathbb{R}^d\)
3: loop
4: if \(\bar{x}\) does not belong to \text{proj}_{\text{bin}} P then
5: \textsc{RandWalkSAT}_\ell(\(\bar{x}\))
6: else
7: \((\text{Output feasible lift of } \bar{x})\) Find \(\bar{y} \in \mathbb{R}^d\) such that \((\bar{x}, \bar{y}) \in P\), return \((\bar{x}, \bar{y})\)
8: end if
9: end loop

As mentioned in the introduction, we show that this algorithm find a feasible solution if such exists, and the running-time improves with the sparsity and decomposability of the instance. Recall the definition of a decomposable mixed-binary problem from equation (1), and let \text{certSupp}_i denote the maximum support size of a minimal projected certificate for the instance \(P^I_i\) which consists only of the \(i\)th block.
Theorem 3.3 (Theorem 2.1 restated). Consider a feasible decomposable mixed-binary set as in equation (1). Then with probability at least $1 - \delta$, mbWalkSAT with parameter $\ell = 1$ returns a feasible solution within $T = \lceil \ln(k/\delta) \rceil \sum_i n_i 2^{n_i \log \text{certSupp}}$ iterations.

In light of Lemma 3.2, if each constraint in $P_i$ has at most $s_i$ integer variables, we have $\text{certSupp} \leq \min\{s_i \cdot (d_i + 1), n_i\}$, and thus this statement indeed implies Theorem 2.1 stated in the introduction. We remark that similar guarantees can be obtained for general $\ell$, but we focus on the case $\ell = 1$ to simplify the exposition.

The high-level idea of the proof of Theorem 3.3 is the following:

1. First we show that if we run mbWalkSAT over a single block $P_i^t$, then with high probability the algorithm returns a feasible solution within $n_i 2^{n_i \log \text{certSupp}} \cdot \ln(1/\delta)$ iterations. This analysis is inspired by the one given by Schöning [Sch99] and argues that with a small, but non-zero, probability the iteration of the algorithm makes the iterate $\bar{x}$ closer (in Hamming distance) to a fixed solution $x^*$ for the instance.

2. Next, we show that when running mbWalkSAT over the whole decomposable instance each iteration only depends on one of the blocks $P_i^t$; this uses the minimality of the certificates. So in effect the execution of mbWalkSAT can be split up into independent executions over each block, and thus we can put together the analysis from Item 1 for all blocks with a union bound to obtain the result.

For the remainder of the section we prove Theorem 3.3. We start by considering a general mixed-binary set as in equation (2). Given such mixed-binary set $P^I$, we use $\text{certSupp} = \text{certSupp}(P^I)$ to denote the maximum support size of all minimal projected certificates.

Theorem 3.4. Consider the execution of mbWalkSAT over a feasible mixed-binary program as in equation (2). The probability that mbWalkSAT does not find a feasible solution within the first $T$ iterations is at most $(1 - p)^{\lceil T/n \rceil}$, where $p = \text{certSupp}^{-n}$. In particular, for $T = n \cdot 2^{n \log (\text{certSupp})} \cdot \lceil \ln(1/\delta) \rceil$ this probability is at most $\delta$ (this follows from the inequality $(1 - x) \leq e^{-x}$ valid for $x \geq 0$).

Proof. Consider a fixed solution $x^* \in \text{proj}_{\text{bin}} P$. To analyze mbWalkSAT, we only keep track of the Hamming distance of the (random) iterate $\bar{x}$ to $x^*$; let $X_t$ denote this (random) distance at iteration $t$, for $t \geq 1$. If at some point this distance vanishes, i.e. $X_t = 0$, we know that $\bar{x} = x^*$ and thus $\bar{x} \in \text{proj}_{\text{bin}} P$; at this point the algorithm returns a feasible solution for $P^I$.

Fix an iteration $t$. To understand the probability that $X_t = 0$, suppose that in this iteration $\bar{x}$ does not belong to $\text{proj}_{\text{bin}} P$, and let $\pi x \leq \pi_0$ be the minimal projected certificate for it used in RandWalkSAT1. Since the feasible point $x^*$ satisfies the inequality $\pi x \leq \pi_0$ but $\bar{x}$ does not, there must be at least one index $i^*$ in the support of $\pi$ such that $x^*$ and $\bar{x}$ differ. Then if algorithm mbWalkSAT makes a “lucky move” and chooses $I = \{i^*\}$ in Line 3, the modified solution after flipping this coordinate (the next line of the algorithm) is one unit closer to $x^*$ in Hamming distance, hence $X_{t+1} = X_t - 1$. Moreover, since $I$ is independent of $I$, the probability of choosing $I = \{i^*\}$ is $1/|\text{supp}(\pi)| \geq 1/\text{certSupp}$. Therefore, if we start at iteration $t$ and for all the next $X_t$ iterations either the iterate belongs to $\text{proj}_{\text{bin}} P$ or the algorithm makes a “lucky move”, it terminates by time $t + X_t$. Thus, with probability at least $(1/\text{certSupp})^{X_t} \geq (1/\text{certSupp})^n = p$ the algorithm terminates by time $t + X_t \leq t + n$.

To conclude the proof, let $\alpha = \lceil T/n \rceil$ and call iterations $i \cdot n, \ldots, (i + 1) \cdot n - 1$ the $i$-th block of iterations. If the algorithm has not terminated by iteration $i \cdot n - 1$, then with probability at least $p$ it terminates within the next $n$ iterations, and hence within the $i$-th block. Putting these bounds together for all $\alpha$ blocks, the probability that the algorithm does not stop by the end of block $\alpha$ is at most $(1 - p)^\alpha$.

This concludes the proof. \(\square\)

Going back to decomposable problems, we now make formal the claim that minimal projected certificates for decomposable mixed-binary sets do not mix the constraints from different blocks. Notice that projected certificates for a decomposable mixed-binary set as in equation (1) have the form $\sum_i \lambda_i A_i x^i \leq \sum_i \lambda_i b^i$ and $\lambda_i b^i = 0$ for all $i \in [k]$.

Lemma 3.5. Consider a decomposable mixed-integer set as in equation (1). Consider a point $\bar{x} \notin \text{proj}_{\text{bin}} P$ and let $\sum_i \lambda_i A_i x^i \leq \sum_i \lambda_i b^i$ be a minimal projected certificate for $\bar{x}$. Then this certificate...
uses only inequalities from one block $P_j$, i.e. there is $j$ such that $\lambda^i = 0$ for all $i \neq j$. Moreover, $\bar{x}^j \not\in \text{proj}_{\text{bin}} P_j$.

Proof. Let us use the natural decomposition $\bar{x} = (\bar{x}^1, \ldots, \bar{x}^k) \in \{0,1\}^{n_1} \times \ldots \times \{0,1\}^{n_k}$, and call the certificate $(\pi x \leq \pi_0) \triangleq (\sum_i \lambda^i A^i \bar{x}^i \leq \sum_i \lambda^i b^i)$. By definition of projected certificate we have $\sum_i \lambda^i A^i \bar{x}^i > \sum_i \lambda^i b^i$, and thus by linearity there must be an index $j$ such that $\lambda^j A^j \bar{x}^j > \lambda^j b^j$. Moreover, as remarked earlier, decomposability implies that the certificate satisfies $\lambda^j B^j = 0$ for all $i$, so in particular for $j$. Thus, the inequality $\lambda^j(A^j, B^j)(\bar{x}^j, y^j) \leq \lambda^j b^j$ obtained by combining only the inequalities from $P_j$ is a projected certificate for $\bar{x}$. The minimality of the original certificate $\pi x \leq \pi_0$ implies that $\lambda^i = 0$ for all $i \neq j$. This concludes the first part of the proof.

Moreover, since $\lambda^j A^j \bar{x}^j > \lambda^j b^j$ and $\lambda^j B^j = 0$ we have that $\lambda^j(A^j, B^j)(\bar{x}^j, y^j) > \lambda^j b^j$ for all $y$, and hence $\bar{x}^j$ does not belong to $\text{proj}_{\text{bin}} P_j$. This concludes the proof.

We can finally prove the desired theorem.

Proof of Theorem 3.3. Again we use the natural decomposition $\bar{x} = (\bar{x}^1, \ldots, \bar{x}^k) \in \{0,1\}^{n_1} \times \ldots \times \{0,1\}^{n_k}$ of the iterates of the algorithm. From Lemma 3.5, we have that, for each scenario, each iteration of mbWalkSAT is associated with just one of the blocks $P^i_j$’s, namely the $P^i_j$ containing all the inequalities in the minimal projected certificate used in this iteration; let $J_i \in [k]$ denote the (random) index $j$ of the block associated to iteration $t$. Notice that at iteration $t$, only the binary variables $x^j$ can be modified by the algorithm.

Let $T_i = n_i 2^{n_i \log n_i} [\ln(k/\delta)]$. Applying the proof of Theorem 3.4 to the iterations $\{\ell : J_{\ell} = i\}$ with index $i$, we get that with probability at least $1 - \delta/k$ the algorithm finds some $x^i$ in $\text{proj}_{\text{bin}} P_i$ within the first $T_i$ of these iterations. Moreover, after the algorithm finds such a point, it does not change it (that is, the remaining iterations have index $J_\ell \neq i$, due to the second part of Lemma 3.5).

Therefore, by taking a union bound we get that with probability at least $1 - \delta$, for all $i \in [k]$ the algorithm finds $x^i \in \text{proj}_{\text{bin}} P_i$ within the first $T_i$ iterations with index $i$ (for a total of $\sum_i T_i = T$ iterations). When this happens, the total solution $x$ belongs to $\text{proj}_{\text{bin}} P$ and the algorithm returns. This concludes the proof.

4 Randomization step RandWalkSAT$_\ell$ within Feasibility Pump

In this section we incorporate the randomization step RANDWALKSAT$_\ell$ into the Naïve Feasibility Pump, the resulting algorithm being called WFP. We describe this algorithm in a slightly different way and using a notation more convenient for the analysis.

Consider a mixed-binary set $P^i$ as in equation (2). Given a 0/1 point $\bar{x} \in \{0,1\}^n$, let $\ell_1$-proj$(P, \bar{x})$ denote a point $(x, y)$ in $P$ where $\| \bar{x} - x \|_1$ is as small as possible. In case of ties, we define $\ell_1$-proj to have the following property.

**Property 4.1** ($\ell_1$-projection gives vertex). Consider a point $\bar{x} \in \{0,1\}^n$ not in $\text{proj}_{\text{bin}} P$, and let $(\bar{x}, y) = \ell_1$-proj$(P, \bar{x})$. Then $\bar{x}$ is a vertex of $\text{proj}_{\text{bin}} P$.

Indeed notice that since $\text{proj}_{\text{bin}} P \subseteq [0,1]^n$ and $\ell_1$-proj$(P, \bar{x})$ is a linear programming problem whose objective function only depends on the $x$-components, there is always a vertex $x$ of $\text{proj}_{\text{bin}} P$ where $(x, y)$ satisfies the desired properties of $\ell_1$-proj$(P, \bar{x})$ for some $y \in \mathbb{R}^d$.

Also, for a vector $v \in [0,1]^n$, we use round$(v)$ to denote the vector obtained by rounding each component of $v$ to the closest integer. We use the convention that $\frac{1}{2}$ is rounded to 1, though any consistent rounding would suffice.

**Property 4.2.** Consider a vector $\bar{x} \in [0,1]^n$. If $\bar{x}_i = \frac{1}{2}$, then round$(\bar{x})_i = 1$.

Notice that operations ‘$\ell_1$-proj’ and ‘round’ correspond precisely to Steps 5 and 4 in the Naïve Feasibility Pump. With this notation, algorithm WFP can be described as follows.
Algorithm 4 WFP

1: **input parameter**: integer \( \ell \geq 1 \)
2: Let \((\bar{x}^0, \bar{y}^0)\) be an optimal solution of the LP relaxation
3: Let \(\bar{x}^0 = \text{round}(\bar{x}^0)\)
4: for \(t = 1, 2, \ldots\) do
5: \((\bar{x}^t, \bar{y}^t) = \ell_1\text{-proj}(P, \bar{x}^{t-1})\)
6: \(\bar{x}^t = \text{round}(\bar{x}^t)\)
7: if \(\bar{x}^t \in \text{proj}_{1\text{in}}(P)\) then
8: Return any \((\bar{x}^t, \bar{y}^t) \in P\)
9: end if
10: if \(\bar{x}^t = \bar{x}^{t-1}\) then \(\triangleright\) iterations have stalled
11: \(\bar{x}^t = \text{RANDWALKSAT}_\delta(\bar{x}^t)\)
12: end if
13: end for

(In Step 7 it suffices to test whether \((\bar{x}^t, \bar{y}^t) \in P\) and return this point: this is because whenever we get \(\bar{x}^t \in \text{proj}_{1\text{in}}(P)\), in the next iteration the projection step will compute \((\bar{x}^{t+1}, \bar{y}^{t+1}) \in P\) with the same 0/1 part \(\bar{x}^{t+1} = \bar{x}^t\), which stays the same after rounding, and thus \((\bar{x}^{t+1}, \bar{y}^{t+1}) \in P\). Also, since RANDWALKSAT was defined over linear inequalities, we think of any equation present in an instance as two opposing inequalities.)

Note that stalling in the above algorithm is determined using the condition \(\bar{x}^t = \bar{x}^{t-1}\). In principle, there could be ‘long cycle’ stalling, that is, \(\bar{x}^t = \bar{x}^{t'}\) where \(t' < t - 1\) but \(\bar{x}^{t'}, \ldots, \bar{x}^{t-1}\) are all distinct binary vectors. As it turns out (assuming no numerical errors) a consistent rounding rule implies that stalling will always occur with cycles of length two.

**Theorem 4.3.** With consistent rounding, long cycle stalling cannot occur.

We present a proof of Theorem 4.3 in Appendix B (also see [GMSS17]).

For the remainder of the section, we analyze the behavior of algorithm WFP on decomposable 1-row instances, proving Theorem 2.4 stated in the introduction. Notice that the projection operators ‘\(\ell_1\text{-proj}\)’ and ‘round’ now present also act on each block independently, namely given a point \(x = (x^1, \ldots, x^k) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k}\), if \((\hat{x}^1, \ldots, \hat{x}^k) = \ell_1\text{-proj}(P, x)\) then \(\hat{x}^i = \ell_1\text{-proj}(P, x^i)\) for all \(i \in [k]\), and similarly for ‘round’. Therefore, as in the proof of Theorem 3.3, it will suffice to analyze the execution of algorithm WFP over a single block/inequality of the decomposable 1-row set. Thus, we start by analyzing these instances, and in Section 4.2 we provide a more formal reduction to this case.

### 4.1 Running time of WFP on 1-row instances

In this section we prove the following guarantee for WFP on a general 1-row instance.

**Theorem 4.4.** Consider a non-empty 1-row set \(P^I = P \cap ([0, 1]^n \times \mathbb{R}^d_+)\) for

\[
P = \{(x, y) \in [0, 1]^n \times \mathbb{R}^d_+: ax + by = c\}.
\]

Then for every \(T \geq 1\), the probability that WFP with \(\ell = 2\) does not find a feasible solution within the first \(T\) iterations is at most \((1 - p)^{\lceil T/(n(n+1)) \rceil}\), where \(p = (1/n^2)^n\). In particular, for \(T = n(n+1) \cdot 2^{2n \log n} \cdot [\ln(1/\delta)]\) this probability is at most \(\delta\).

The high-level idea of the proof of this theorem is the following. We use a similar strategy as before, where we consider a fixed feasible solution \(x^* \in \text{proj}_{1\text{in}} P\) and track its distance to the iterates \(\bar{x}^t\) generated by algorithm WFP. However, while again the randomization step RANDWALKSAT brings \(\bar{x}^t\) closer to \(x^*\) with small but non-zero probability, the issue is that the projections ‘\(\ell_1\text{-proj}\)’ and ‘round’ in the next iterations could send the iterate even further from \(x^*\). To analyze the algorithm we first have to track the distance to a special feasible solution \(x^*\) (namely a coordinate-wise maximal one), use the structure of 1-row instances to carefully analyze the effect of the projections involved, and show that a round of RANDWALKSAT plus ‘\(\ell_1\text{-proj} + \text{round}\)’ still has a non-zero probability of generating a point
closer to $x^*$. For this, it will be actually important that we use $\ell = 2$ in algorithm WFP (in fact, $\ell \geq 2$ suffices).

For the remainder of the section we prove Theorem 4.4. To simplify the notation we omit the polytope $P$ from the notation of $\ell_1$-proj. Given a point $\tilde{x} \in \{0,1\}^n$, let $\text{AltProj}(\tilde{x})$ be the effect of applying to $\tilde{x}$ the function $\ell_1\text{-proj}(\cdot)$ and then round($\cdot$), namely if $(x, y) = \ell_1\text{-proj}(\tilde{x})$ then $\text{AltProj}(\tilde{x}) = \text{round}(\tilde{x})$. Notice this is again a 0/1 vector; moreover, if $\tilde{x}$ belongs to $\text{proj}_{\text{bin}} P$, then $\text{AltProj}(\tilde{x}) = \tilde{x}$. Then algorithm WFP can be thought as performing a AltProj operation, then checking if the iterate obtained either belongs to $\text{proj}_{\text{bin}} P$ (in which case it exits) or if it equals the previous iterate (in which case it applies $\text{RandWalkSAT}_\ell$); if neither of these occur, then another AltProj operation is performed.

It will be then convenient to compress a sequence of operations AltProj into its “closure” AltProj. More precisely, define the iterated operation $\text{AltProj}^t(\tilde{x}) = \text{AltProj}\left(\text{AltProj}^{t-1}(\tilde{x})\right)$ (with $\text{AltProj}^1 = \text{AltProj}$), and if the sequence $(\text{AltProj}^t(\tilde{x}))_t$ stabilizes at a point, let $\text{AltProj}^*(\tilde{x})$ denote this point. We then arrive at the compressed version of the algorithm WFP.

Algorithm 5 WFP-Compressed

1. input parameter: integer $\ell \geq 1$
2. Let $(\tilde{x}^0, \tilde{y}^0)$ be an optimal solution of the LP relaxation
3. Let $z^0 = \text{round}(\tilde{x}^0)$
4. for $\tau = 1, 2, \ldots$ do
5. $\tilde{z}^\tau = \text{AltProj}^*(z^{\tau-1})$
6. if $\tilde{z}^\tau \in \text{proj}_{\text{bin}} P$ then
7. Return $\tilde{z}^\tau$
8. end if
9. $\tilde{z}^\tau = \text{RandWalkSAT}_{\ell}(\tilde{z}^\tau)$
10. end for

Thus, WFP-Compressed starts with a point $\tilde{z}^1$ and repeatedly applies the operation $\text{AltProj}(\text{RandWalkSAT}_{\ell}(\cdot))$ to obtain the sequence $\tilde{z}^1, \tilde{z}^2, \ldots$ until one of these terms belongs to $\text{proj}_{\text{bin}} P$.

By using the same randomness in both WFP and WFP-compressed (or more precisely coupling the outcomes of $\text{RandWalkSAT}_{\ell}(\cdot)$ on both algorithms) we see that in all scenarios the number of iterations WFP and WFP-Compressed is related by how large a sequence of AltProj’s we can have before stabilizing:

$$\# \text{ iterations WFP} \leq \left[\# \text{ iterations WFP-Compressed}\right] \cdot \max_{\tilde{x} \in \{0,1\}^n} \min \{k : \text{AltProj}^k(\tilde{x}) = \text{AltProj}^*(\tilde{x})\}.$$  

(4)

Thus, from now on we focus on analyzing the number of iterations WFP-Compressed (with $\ell = 2$) takes and in controlling the multiplicative factor in this inequality.

In the next few lemmas, we start by understanding the behavior of AltProj alone. First, some basic properties related to the $\ell_1$-projection it preforms.

Lemma 4.5. The following hold:

1. The set $\text{proj}_{\text{bin}} P$ is equal to either the set $[0,1]^n$, the set $\{x \in [0,1]^n : ax \leq c\}$, the set $\{x \in [0,1]^n : ax \geq c\}$, or the set $\{x \in [0,1]^n : ax = c\}$.
2. For any point $\tilde{x} \in \{0,1\}^n$, $\ell_1\text{-proj}(\tilde{x})$ has at most one fractional coordinate.
3. For any point $\tilde{x} \in \{0,1\}^n$ not in $\text{proj}_{\text{bin}} P$, or equivalently $\|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 > 0$, we have a $\ell_1\text{-proj}(\tilde{x}) = c$.

Proof. (Item 1.) It is immediate that $\text{proj}_{\text{bin}}$ depends on the coefficients $b$ of the continuous variables in the following way: $\text{proj}_{\text{bin}} P$ is equal to $[0,1]^n$ if $b$ has a positive and a negative coefficient, equal to $\{x \in [0,1]^n : ax = c\}$ if $b = 0$, equal to $\{x \in [0,1]^n : ax \leq c\}$ if $b \geq 0$ and it has a positive coefficient, or equal to $\{x \in [0,1]^n : ax \geq c\}$ if $b \leq 0$ and it has a negative coefficient. Notice these cover all the cases for the possible sign combinations in $b$.  

9
(Item 2.) Recall \( \ell_1\text{-proj}(\tilde{x}) \) is a vertex of \( \text{proj}_{\bin}$$. But since \( \text{proj}_{\bin} \) only has one equation/inequality in addition to the bounds \([0, 1]^n \), it is well-known that its vertices (or equivalently basic feasible solutions) will have all but at most one of its coordinates set to its upper or lower bound; thus, all but possibly this special coordinate have 0/1 value.

(Item 3.) The intuition if \( \tilde{x} \notin \text{proj}_{\bin} \), then the \( \ell_1 \)-projection of this point to \( \text{proj}_{\bin} \) lies on the boundary of \( \text{proj}_{\bin} \), and thus should satisfy the equality \( ax = c \).

More precisely, let \( \tilde{x} = \ell_1\text{-proj}(\tilde{x}) \). Recall the classification of \( \text{proj}_{\bin} \) from Item 1. If \( \text{proj}_{\bin} = [0, 1]^n \) then \( \tilde{x} \in \text{proj}_{\bin} \), and if \( \text{proj}_{\bin} = \{x \in [0, 1]^n : ax \leq c\} \) then clearly \( ax = c \). So assume \( \text{proj}_{\bin} = \{x \in [0, 1]^n : ax \leq c\} \) (the case \( ax \geq c \) is analogous). By contradiction suppose \( \tilde{x} \notin \text{proj}_{\bin} \), so \( a\tilde{x} > c \), but \( a\tilde{x} < c \). Then there is \( \varepsilon > 0 \) such that \( \varepsilon\tilde{x} + (1 - \varepsilon)x \) belongs to \( \text{proj}_{\bin} \). But this point is closer in \( \ell_1 \) to \( x \) than \( \tilde{x} \) is, contradicting the minimality of the latter. This concludes the proof.

The following is the starting point for understanding when a sequence of \text{AltProj}'s stabilizes.

**Lemma 4.6.** Consider a point \( \tilde{x} \in [0, 1]^n \). Then:

1. If \( \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 < \frac{1}{2} \), then \( \text{AltProj}(\tilde{x}) = \tilde{x} \).
2. If \( \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 = \frac{1}{2} \), then \( \text{AltProj}(\tilde{x}) \) is coordinate-wise at least \( \tilde{x} \) and these vectors differ in at most one coordinate.

**Proof.** Consider a point \( \tilde{x} \in [0, 1]^n \) with \( \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 \in (0, \frac{1}{2}] \) (in case \( \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 = 0 \) Item 1 clearly holds). This implies that the vectors \( \ell_1\text{-proj}(\tilde{x}) \) and \( \tilde{x} \) differ exactly in the unique (by the lemma above) fractional coordinate of \( \tilde{x} \); let \( j \) denote this coordinate. This implies that after rounding we have \( \text{AltProj}(\tilde{x}) = \text{round}(\ell_1\text{-proj}(\tilde{x})) \) for all \( i \neq j \).

This also implies that \( \|\ell_1\text{-proj}(\tilde{x})_j - \tilde{x}_j\|_1 = \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 \). If the right-hand side is strictly less than \( \frac{1}{2} \), we have \( \text{round}(\ell_1\text{-proj}(\tilde{x}))_j = \tilde{x}_j \), and thus \( \text{round}(\ell_1\text{-proj}(\tilde{x})) = \tilde{x} \); this proves Item 1 of the lemma. If instead we have \( \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 = \frac{1}{2} \), then \( \ell_1\text{-proj}(\tilde{x})_j \) must be equal to \( \frac{1}{2} \), which implies that \( \text{round}(\ell_1\text{-proj}(\tilde{x}))_j = 1 \). Thus, \( \text{AltProj}(\tilde{x})_j \geq \tilde{x}_j \), and since this is the only coordinate where these vectors can differ we have the proof of Item 2 of the lemma.

The next lemma shows that regardless of the starting point, after only one application of \text{AltProj} we end up in one of the cases of the lemma above.

**Lemma 4.7.** Consider \( \tilde{x} \in [0, 1]^n \) and let \( \tilde{x}' = \text{AltProj}(\tilde{x}) \). Then \( \|\ell_1\text{-proj}(\tilde{x}') - \tilde{x}'\|_1 \leq \frac{1}{2} \).

**Proof.** Let \( \tilde{x} = \ell_1\text{-proj}(\tilde{x}) \) and recall \( \tilde{x}' = \text{round}(\tilde{x}) \). Since \( \tilde{x} \) has at most one fractional coordinate (Lemma 4.5), this is the only one that can be rounded (to the nearest integer) and hence \( \|\tilde{x} - \tilde{x}'\|_1 \leq \frac{1}{2} \).

Since \( \ell_1\text{-proj}(\tilde{x}') \) is a minimizer of \( \min_{x \in \text{proj}_{\bin}} \|x - \tilde{x}'\|_1 \) and \( \tilde{x} \) is a feasible solution for this minimization problem, we have \( \|\ell_1\text{-proj}(\tilde{x}') - \tilde{x}'\|_1 \leq \frac{1}{2} \). This concludes the proof.

Since after the first application of \text{AltProj} we satisfy the conditions of Lemma 4.6, and since this lemma guarantees that each further applications of \text{AltProj} either does not do anything or component-wise increases the input vector (and we cannot have more than \( n \) of such increases), we get that we stabilize after at most \( n + 1 \) applications of \text{AltProj}.

**Corollary 4.8.** For any point \( \tilde{x} \in [0, 1]^n \) \( \text{AltProj}^n(\tilde{x}) = \text{AltProj}^{n+1}(\tilde{x}) \).

In particular, this shows that the last term in the right-hand side of (4) is at most \( n + 1 \).

To be able to analyze the effect that \text{RandWalkSAT}_2, when combined with \text{AltProj}, we need to obtain a finer understanding of the case of Item 2 in Lemma 4.6, namely when there are multiple applications of \text{AltProj} before stabilizing. The following example is very representative of when this happens.

**Example 4.9.** Consider the pure-integer 1-row set with left-hand side coefficients \( a = (2, -2, 2, 1) \) and right-hand side \( c = 1 \), so its relaxation is

\[ P = \{x \in [0, 1]^4 : 2x_1 - 2x_2 + 2x_3 + x_4 = 1\} \]

Notice that any feasible solution sets \( x_4 = 1 \).
Now consider starting at the point \( \vec{x} = (0, 0, 0, 0) \) and the sequence of iterates \((\text{AltProj}^j(\vec{x}))_j\). In the first step we have two options for the projection \( \ell_1\text{-proj}(\vec{x}) = (0, 0, 0, 0) \) (due to the symmetry between \(x_1\) and \(x_2\)), so for concreteness assume \( \ell_1\text{-proj}(\vec{x}) = (\frac{1}{2}, 0, 0, 0) \); notice this falls on Item 2 of Lemma 4.6. After rounding we then have \( \text{AltProj}(\vec{x}) = (1, 0, 0, 0) \). In the next step, when projecting \( \text{AltProj}(\vec{x}) \) we again have two options (due now to a different symmetry between \(x_1\) and \(x_2\)), so for concreteness assume \( \ell_1\text{-proj}(\text{AltProj}(\vec{x})) = (1, \frac{1}{2}, 0, 0, 0) \) (again Item 2 of Lemma 4.6); thus, \( \text{AltProj}^2(\vec{x}) = (1, 1, 0, 0) \). Proceeding in this way, we may obtain \( \text{AltProj}^j(\vec{x}) = (1, 1, 1, 0) \), at which point the sequence finally stabilizes: \( \text{AltProj}^j(\vec{x}) = \text{AltProj}^j(\vec{x}) \), which then equals \( \text{AltProj}^j(\vec{x}) \).

The next lemma shows that the only way we can have a long sequence of AltProj before stabilizing is when coordinates of the left-hand side of opposite signs are being “added” to our iterates (e.g. the 2’s and \(-2\)’s in the above example).

**Lemma 4.10.** Consider a point \( \vec{x} \in \{0,1\}^n \) with \( \|\ell_1\text{-proj}(\vec{x}) - \vec{x}\|_1 \leq \frac{1}{4} \). Consider the points \( \vec{x}' = \text{AltProj}(\vec{x}) \) and \( \vec{x}'' = \text{AltProj}^2(\vec{x}) \). Suppose \( \vec{x} \neq \vec{x}' \neq \vec{x}'' \), and from Lemma 4.6(2) let \( i_1, i_2 \in [n] \) be the indices such that \( \supp(\vec{x}') = \supp(\vec{x}) \cup \{i_1\} \) and \( \supp(\vec{x}'') = \supp(\vec{x}') \cup \{i_2\} \). Then \( a_{i_1} = -a_{i_2} \).

**Proof.** To simplify the notation define \( \vec{x} = \ell_1\text{-proj}(\vec{x}) \) and \( \vec{x}' = \ell_1\text{-proj}(\vec{x}') \). Since \( \vec{x}' = \vec{x} + e^{i_1} \), \( \|\vec{x} - \vec{x}\|_1 = \frac{1}{2} \), and round(\( \vec{x} \)) = \( \vec{x}' \), we have \( \vec{x} = \vec{x}_i = \frac{1}{2} \) and \( \vec{x}_i = \vec{x}_i \) for all \( i \neq i_1 \). Thus \( \vec{x} = \vec{x} + \frac{1}{2} e^{i_1} \).

Lemma 4.7 guarantees that \( \|\vec{x}' - \vec{x}\|_1 = \frac{1}{2} \), and because \( \vec{x}' \neq \vec{x}'' \) Lemma 4.6(1) guarantees that actually \( \|\vec{x}' - \vec{x}\|_1 = \frac{1}{2} \). Thus, the same argument above holds with \( \vec{x}' \) replaced for \( \vec{x} \) and gives that \( \vec{x}' = \vec{x} + \frac{1}{2} e^{i_1} \). In particular, \( \vec{x} - \vec{x}' = \frac{1}{2} (e^{i_1} + e^{i_1}) \).

From Lemma 4.5(3) we have that the points \( x \) and \( \vec{x} \) satisfy \( ax = c \). Thus, taking their differences and using the equality above we obtain

\[
0 = a(\vec{x} - \vec{x}') = -\frac{1}{2} a(e^{i_1} + e^{i_1}) = -\frac{1}{2} (a_{i_1} + a_{i_2}),
\]

which implies \( a_{i_1} = -a_{i_2} \). This concludes the proof. \( \square \)

Finally we start bringing \( \text{RandWalksSAT}_2 \) to the picture. The next lemma shows that given any point \( \vec{x} \), there is a “lucky choice” in \( \text{RandWalksSAT}_2(\vec{x}) \) that changes at most two coordinates of the vector and brings us closer to a feasible solution \( x^* \). Importantly, it also gives us precise control on the \( \ell_1\)-projection of the obtained point, which will be crucial for analyzing the effect of applying AltProj\(^*\) to the new point obtained.

**Lemma 4.11.** Consider a point \( x^* \in \{0,1\}^n \) in \( \text{proj}_{\text{bin}} P \), and a point \( \vec{x} \in \{0,1\}^n \) not in \( \text{proj}_{\text{bin}} P \). Suppose \( \text{AltProj}(\vec{x}) = \vec{x} \). Then there is a point \( \vec{x}' \in \{0,1\}^n \) satisfying the following:

1. (close to \( \vec{x} \)) \( \|\vec{x}' - \vec{x}\|_1 \leq 2 \)
2. (closer to \( x^* \)) \( \|\vec{x}' - x^*\|_1 \leq \|\vec{x} - x^*\|_1 - 1 \)
3. (projection control) \( \|\ell_1\text{-proj}(\vec{x}') - \vec{x}\|_1 \leq \frac{1}{2} \).

Moreover, if we have the equality \( \|\ell_1\text{-proj}(\vec{x}') - \vec{x}\|_1 = \frac{1}{2} \) in Item 3, then \( \|\vec{x}' - x^*\|_1 \leq \|\vec{x} - x^*\|_1 - 2 \).

**Proof.** Recall the classification of \( \text{proj}_{\text{bin}} P \) from Lemma 4.5. Since the 0/1 point \( \vec{x} \) does not belong to \( \text{proj}_{\text{bin}} P \), we cannot have \( \text{proj}_{\text{bin}} P = \{0,1\}^n \). Let us consider the other possible cases and understand the relations \( ax^* \preceq c \) and \( a\vec{x} \succeq c \) that come from the assumption on these points.

If \( \text{proj}_{\text{bin}} P \) is in the “less-than-or-equal” case, i.e. \( \text{proj}_{\text{bin}} = \{x \in [0,1]^n : ax \leq c\} \), then we have \( ax^* \leq c \) and \( a\vec{x} > c \); if we are in the “greater-than-or-equal” case \( \text{proj}_{\text{bin}} = \{x \in [0,1]^n : ax \geq c\} \), then \( ax^* \geq c \) and \( a\vec{x} < c \); finally, if we are in the “equality” case \( \text{proj}_{\text{bin}} = \{x \in [0,1]^n : ax = c\} \), we have \( ax^* = c \) and \( a\vec{x} \neq c \). Therefore, to consider all these cases together, we can just consider \( ax^* \preceq c \) and \( a\vec{x} \geq c \), where ‘\( \preceq \)’ is either of the strict relations ‘\( \prec \)’ or ‘\( > \)’ is the opposite relation, and ‘\( \succeq \)’ is the predicate ‘\( \preceq \) or =’.

Define \( \chi = \vec{x} - x^* \); so identifying \( \vec{x} \) and \( x^* \) with the corresponding sets they indicate, \( \chi_1 = 1 \) if \( i \) belongs to \( \vec{x} \) but not \( x^* \), \( \chi_1 = -1 \) if \( i \) belongs to \( x^* \) but not \( \vec{x} \), and \( 0 \) otherwise. Thus, to construct a \( \vec{x}' \) that is closer to \( x^* \) than \( \vec{x} \), we will subtract from \( \vec{x} \) one or two terms \( \chi_i \)'s.
Since
\[ 0 < a\tilde{x} - c < a(\tilde{x} - x^*) = \sum_i a_i \chi_i, \] (5)
there is at least one index where \( a_i \chi_i > 0 \); we break into two cases depending on how \( \cdot \cdot \cdot \) big such a value can be.

**Case 1:** There is index \( j \) such that \( \chi_j a_j > 0 \) and \( \chi_j a_j \preceq a\tilde{x} - c \). Then define \( \tilde{x}' = \tilde{x} - \chi_j e_j \). It is clear that this point satisfies Items 1 and 2 of the lemma, so we focus on Item 3. For that, we will construct a candidate \( u \) for the \( \ell_1 \)-projection onto \( \text{proj}_{\binom{0}{1}} P \) that satisfies \( \|u - \tilde{x}\|_1 < \frac{1}{2} \).

Since \( \text{AltProj}(\tilde{x}) = \tilde{x} \), Lemma 4.7 implies \( \|\ell_1\text{-proj}(\tilde{x}) - \tilde{x}\|_1 \leq \frac{1}{2} \). Also, by Lemma 4.5 \( \ell_1\text{-proj}(\tilde{x}) \) has exactly one fractional coordinate, say, coordinate \( k \) (if \( \ell_1\text{-proj}(\tilde{x}) \) has no fractional coordinates then \( \tilde{x} = \text{round}(\ell_1\text{-proj}(\tilde{x})) = \ell_1\text{-proj}(\tilde{x}) \) and so \( \tilde{x} \in \text{proj}_{\binom{0}{1}} P \), contradicting its definition); together these imply that \( \tilde{x} - \ell_1\text{-proj}(\tilde{x}) = \alpha e_k \) for some \( \alpha \in [-1/2, 1/2] \).

We claim that the coordinates \( k \) and \( j \) are different. To see this, notice that
\[
\begin{align*}
ad\tilde{x} - c &= a(\tilde{x} - \ell_1\text{-proj}(\tilde{x})) = a\tilde{a}_k, \\
\text{and since } |\alpha| &\leq \frac{1}{2} \text{ this implies that } |a_k| \geq 2|a\tilde{x} - c|; \text{ by definition of } j, \text{ this is strictly greater than } |a_j|, \\
\text{and thus } j &\neq k.
\end{align*}
\]
So define the point \( u = \tilde{x}' - \beta \alpha e_k \) for some \( \beta \) such that \( au = c \); notice that such \( \beta \) exists and belongs to the interval \((0, 1)\), since at the bounds of this interval we get (using the definition of \( j \))
\[
ad\tilde{x}' = a(\tilde{x} - \chi_j e_j) = a\tilde{x} - a_j > c \]
and
\[
ad(\tilde{x}' - \alpha e_k) = a(\ell_1\text{-proj}(\tilde{x}) - \chi_j e_j) = c - a_j < c, \]
and since \( a(\tilde{x}' - \beta \alpha e_k) \) is continuous on \( \beta \) (so we can use the Intermediate Value Theorem).

Notice that \( u \in \text{proj}_{\binom{0}{1}} P \); by construction \( au = c \), \( u_i \in [0, 1] \) for all \( i \neq k \) (because \( u_i = \tilde{x}'_i \) and the right-hand side belongs to \( \{0, 1\} \)), and also \( u_k \in [0, 1] \) (because we have the convex combination
\[
u_k = (1 - \beta)\tilde{x}_k + \beta(\tilde{x}' - \alpha) = (1 - \beta)\tilde{x}_k + \beta(\tilde{x} - \alpha)
\]
and the terms \( \tilde{x}_k \) and \( \tilde{x}_k - \alpha = \ell_1\text{-proj}(\tilde{x})_k \) belong to \([0, 1]\)). Thus, \( u \) is indeed a candidate for \( \ell_1 \)-projection onto \( \text{proj}_{\binom{0}{1}} P \), and hence
\[
\|\ell_1\text{-proj}(\tilde{x}') - \tilde{x}'\|_1 \leq \|u - \tilde{x}'\|_1 = \beta|\alpha| < \frac{1}{2}.
\]

**Case 2:** There is an index \( j \) with \( a_j \chi_j > 0 \) and \( a_j \chi_j \preceq a\tilde{x} - c \). Given this hypothesis, there must be also an index \( k \) such that \( a_k \chi_k < 0 \), since from equation (5) we have that \( \sum_i a_i \chi_i \preceq a\tilde{x} - c \). To construct the vectors \( \tilde{x}' \) and \( u \), consider the 2-dimensional system on variables \( \alpha, \beta \)
\[
a\tilde{x} - \alpha a_j \chi_j - \beta a_k \chi_k = c
\]
\[
\alpha, \beta \in [0, 1].
\]
This system is feasible, since setting \( (\alpha, \beta) = (0, 0) \) we obtain \( a\tilde{x} - c \) and setting \( (\alpha, \beta) = (1, 0) \) we obtain \( a\tilde{x} - a_j \chi_j < c \) (and as before we have continuity on \( \alpha \)); in fact, this argument shows that there is a solution on the semi-open box \((0, 1) \times [0, 1]\). Moreover, because the terms \( a_j \chi_j > 0 \) and \( a_k \chi_k < 0 \) have opposite signs, the line \( \{ (\alpha, \beta) \in \mathbb{R}^2 : a\tilde{x} - \alpha a_j \chi_j - \beta a_k \chi_k = c \} \) (or more precisely the function \( \beta = \beta(\alpha) \) it represents) has positive slope, and thus intersects the box \([1, 0]^2 \) in either the line \( \{0\} \times [0, 1] \) or the line \([0, 1] \times \{1\} \). Thus, there is a solution \( (\tilde{\alpha}, \tilde{\beta}) \) to the system where at least one of \( \tilde{\alpha} \) or \( \tilde{\beta} \) equals 1.

Now define \( (\tilde{\alpha}, \tilde{\beta}) = \text{round}(\tilde{\alpha}, \tilde{\beta}) \), where again we round \( \frac{1}{2} \) to 1. Then define \( u = \tilde{x} - \tilde{\alpha} \chi_j e_j - \tilde{\beta} \chi_k e_k \) and \( \tilde{x}' = \tilde{x} - \tilde{\alpha} \chi_j e_j - \tilde{\beta} \chi_k e_k \). Clearly \( \tilde{x}' \) satisfies Item 1 of the lemma: \( \|\tilde{x}' - \tilde{x}\|_0 \leq 2 \). Also, as before \( u \) is a candidate for \( \ell_1 \)-projection onto \( \text{proj}_{\binom{0}{1}} P \), which gives the \( \ell_1 \) bound
\[
\|\ell_1\text{-proj}(\tilde{x}') - \tilde{x}'\|_1 \leq \|u - \tilde{x}'\|_1 = \|((\tilde{\alpha}, \tilde{\beta}) - (\tilde{\alpha}, \tilde{\beta}))\|_1.
\]
Recall at least one of \( \tilde{\alpha} \) or \( \tilde{\beta} \) equals 1.
• if the other one equals any value different from $\frac{1}{2}$, the right-hand side of (6) is strictly less than $\frac{1}{2}$ (so Item 3 of the lemma is satisfied) and we still have $\|\tilde{x} - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1$ (because at least one of $\tilde{\alpha}$ or $\tilde{\beta}$ equals 1).

• if instead the other one equals $\frac{1}{2}$, then the right-hand side of (6) is equals $\frac{1}{2}$ but both $\tilde{\alpha}$ and $\tilde{\beta}$ equal 1, thus $\|\tilde{x} - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 2$.

This concludes the proof.

Now take the point $\tilde{x}'$ closer to $x^*$ than $\tilde{x}$ constructed in the previous lemma and consider the effect of applying AltProj$^\ast$ to $\tilde{x}'$. We would like to obtain that the final point AltProj$(\tilde{x}')$ is still closer to $x^*$ than where we began, i.e., we would like $\|\text{AltProj}^\ast(\tilde{x}') - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1$. If AltProj$^\ast(\tilde{x}') = \tilde{x}'$ this is clearly true, but if this does not happen we know (for instance from Lemma 4.6) that the repeated application of AltProj can only coordinate-wise increase the vector $\tilde{x}'$. Thus, if we compare the final vector AltProj$^\ast(\tilde{x}')$ with a maximal feasible solution $x^*$ there is a chance that these applications of AltProj did not take us further from $x^*$. In order to make this very crude intuition precise we need to use the finer control on the effect of AltProj given by Lemma 4.10 and the extra slack in the guarantee $\|\tilde{x}' - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 2$ of the lemma above when $\|\ell_1 - \text{proj}(\tilde{x}') - \tilde{x}'\|_1 = \frac{1}{2}$.

**Lemma 4.12.** Let $x^\ast$ be a coordinate-wise maximal solution in $\{0,1\}^n \cap \text{proj}_{\text{bin}} P$. Consider any point $\tilde{x} \in \{0,1\}^n \cap \text{proj}_{\text{bin}} P$ satisfying $\text{AltProj}(\tilde{x}) = \tilde{x}$, and let $\tilde{x}' \in \{0,1\}^n$ be a point constructed in Lemma 4.11 with respect to $x^\ast$ and $\tilde{x}$. Then $\|\text{AltProj}^\ast(\tilde{x}') - x^\ast\|_0 \leq \|\tilde{x} - x^\ast\|_0 - 1$.

**Proof.** If AltProj$(\tilde{x}') = \tilde{x}'$ then the result holds, since by definition $\|\tilde{x}' - x^\ast\|_0 \leq \|\tilde{x} - x^\ast\|_0 - 1$. So suppose AltProj$(\tilde{x}') \neq \tilde{x}'$; since $\|\ell_1 - \text{proj}(\tilde{x}') - \tilde{x}'\| \leq \frac{1}{2}$ and Lemma 4.6 precludes this inequality holds strictly, we have $\|\ell_1 - \text{proj}(\tilde{x}') - \tilde{x}'\|_1 = \frac{1}{2}$. Thus, we get that stronger guarantee in Lemma 4.11 that $\|\tilde{x}' - x^\ast\|_0 \leq \|\tilde{x} - x^\ast\|_0 - 2$.

Now let $k$ be the smallest such that AltProj$^\ast(\tilde{x}') = \text{AltProj}^\ast(\tilde{x}')$, which exists from Corollary 4.8; so we want to show $\|w^t - x^\ast\|_0 \leq \|\tilde{x} - x^\ast\|_0 - 1$. To simplify the notation, let $w^t \triangleq \text{AltProj}^\ast(\tilde{x}')$ for $t = 0, 1, \ldots, k$. From Lemma 4.7 we have that $\|\ell_1 - \text{proj}(w^t) - w^t\| \leq \frac{1}{2}$ for all $t$. Then the characterization of sequences of AltProj$^\ast$ of Lemma 4.10 give that there are indices $i_1, \ldots, i_k$ such that

$$w^{t-1} + e^{i_t} = w^t \quad t = 1, 2, \ldots, k$$

and that satisfy the alternating relation $a_{i_t} = -a_{i_{t+1}}$ for all $t = 1, 2, \ldots, k - 1$. Thus, the sequence $(a_{i_t})_t$ only contains the values $v$ and $-v$ for some $v \in \mathbb{R}$.

Notice that since the $i_t$'s do not belong to the support of $w^0$, we see (for instance by induction) that

$$\|w^k - x^\ast\|_0 = \|w^0 - x^\ast\|_0 - \# t \text{'s with } i_t \in \text{supp}(x^\ast) + \# t \text{'s with } i_t \notin \text{supp}(x^\ast).$$

(7)

But notice that the values $v$ and $-v$ cannot both be outside supp$(x^\ast)$, i.e., there are no indices $i, j \notin \text{supp}(x^\ast)$ with $a_i = v$ and $a_j = -v$, otherwise we could add them to $x^\ast$ (i.e., consider $x^\ast + e^i + e^j$) and obtain a coordinate-wise larger point in $\{0,1\}^n \cap \text{proj}_{\text{bin}} P$, contradicting the maximality of $x^\ast$. Thus, we obtain that roughly at most half of the $i_t$'s belong are outside supp$(x^\ast)$:

$$\# t \text{'s with } i_t \notin \text{supp}(x^\ast) \leq \left\lfloor \frac{k}{2} \right\rfloor$$

(and the rest of the $i_t$'s belong to supp$(x^\ast)$). Employing these bounds to equation (7) we get

$$\|w^k - x^\ast\|_0 \leq \|w^0 - x^\ast\|_0 - \left\lfloor \frac{k}{2} \right\rfloor + \left\lfloor \frac{k}{2} \right\rfloor \leq \|w^0 - x^\ast\|_0 + 1.$$

But as mentioned in the beginning of the proof $\|w^0 - x^\ast\|_0 = \|\tilde{x} - x^\ast\|_0$ is at most $\|\tilde{x} - x^\ast\|_0 - 2$; thus, $\|w^k - x^\ast\|_0 \leq \|\tilde{x} - x^\ast\|_0 - 1$, which concludes the proof.

Now going back to algorithm WFP-Compressed. Notice that since $\tilde{x}^\ast$ is obtained from AltProj$^\ast(\cdot)$, it satisfies the fixed point condition AltProj$(\tilde{x}^\ast) = \tilde{x}^\ast$. Thus, as long as $\tilde{x}^\ast$ does not belong to proj$_{\text{bin}} P$ we can apply the above lemma to obtain that with probability at least $\frac{1}{17}$ the procedure RANDWALKSAT$_2$ will flip coordinates of $\tilde{x}^\ast$ in a way that $\tilde{x}^{\ast+1} = \text{AltProj}^\ast(\text{RANDWALKSAT}_2(\tilde{x}^\ast))$ is closer to $x^\ast$ in $\ell_0$ than the previous iterate $\tilde{x}^\ast$. 

13
Corollary 4.13. Let \( x^* \) be a coordinate-wise maximal point in \( \{0,1\}^n \cap \text{proj}_{\text{bin}} P \). Then
\[
\Pr\left( \|\tilde{z}^{t+1} - x^*\|_0 \leq \|\tilde{z}^t - x^*\|_0 - 1 \mid \tilde{z}^t \notin P \right) \geq \frac{1}{n^2}.
\]

Now we can conclude the proof of Theorem 4.4 arguing just like in the proof of Theorem 3.4.

Proof of Theorem 4.4. We bound the number of iterations of algorithm WFP-Compressed first. Fix \( T \geq 1 \) and let \( T' = T/(n + 1) \).

Let \( x^* \) be a coordinate-wise maximal point in \( \{0,1\}^n \cap \text{proj}_{\text{bin}} P \), and let \( Z_{\tau} = \|\tilde{z}^\tau - x^*\|_0 \) Notice that \( Z_{\tau} = 0 \) implies \( \tilde{z}^\tau = x^* \) and hence \( \tilde{z}^\tau \in P \), which implies that the algorithm stops. Corollary 4.13 gives that \( \Pr(\tilde{Z}_{\tau+1} \leq Z_{\tau} - 1 \mid \tilde{z}^\tau \notin P) \geq \frac{1}{n^2} \). Therefore, if we start at iteration \( \tau \) and for all the next \( Z_{\tau} \) iterations either the iterate \( \tilde{z}^\tau \) belongs to \( P \) or the algorithm reduces \( Z_{\tau} \), it terminates by time \( \tau + Z_{\tau} \). Thus, with probability at least \( (1/n^2)^{T'} \geq (1/n^2)^n = p \) the algorithm terminates by time \( \tau + Z_{\tau} \) within any of the \( \alpha \) blocks.

Now let \( \alpha = [T'/n] \) and call time steps \( i \cdot n, \ldots, (i + 1) \cdot n - 1 \) the \( i \)-th block of time. From the above paragraph, the probability that there is \( \tau \) in the \( i \)-th block of time such that \( \tilde{z}^\tau \in P \) conditioned on \( \tilde{z}^{n-1} \notin P \) is at least \( p \). Using the chain rule of probability gives that the probability that there is no \( \tilde{z}^\tau \in P \) within any of the \( \alpha \) blocks is at most \( (1 - p)^\alpha \). This shows that with probability at least \( 1 - (1 - p)^\alpha \), algorithm WFP-Compressed terminates after at most \( T' \) iterations.

Moreover, since from Corollary 4.8 we have that \( \text{AltProj}_{n+1}(\tilde{x}) = \text{AltProj}_n(\tilde{x}) \), it follows from inequality (4) that the original algorithm WFP terminates in at most \( T' \cdot (n + 1) = T \) iterations with probability at least \( 1 - (1 - p)^\alpha = 1 - (1 - p)^{T/(n(n+1))} \). This concludes the proof.

\( \square \)

4.2 Proof of Theorem 2.4

Fix a decomposable 1-row set \( P \) and let \( P_i \) denote its \( i \)-th block, so \( P = P_1 \times P_2 \times \ldots \times P_k \). Consider the execution of algorithm WFP over \( P \), and let \( \tilde{z}^\tau \in \{0,1\}^{n_1 + \ldots + n_k} \) be the iterate produced by WFP at the end of iteration \( t \). Let \( \text{proj}_i : \mathbb{R}^{n_1 + \ldots + n_k} \rightarrow \mathbb{R}^{n_i} \) denote the canonical projection to the coordinates corresponding to the \( i \)-th block of \( P \) (so \( \text{proj}_i \tilde{z}^\tau \) is the binary part of a tentative solution for the \( i \)-th block).

As in the proof of Theorem 3.3, from Lemma 3.5 we have that, for each scenario, each application of \( \text{RandWalkSAT}_i \) acts on only one block of \( P \), namely the \( P_i \) containing the inequality comprising the minimal projected certificate used. If operator \( \text{RandWalkSAT}_i \) is invoked in iteration \( t \) of WFP, let \( J_i \in [k] \) denote the (random) index \( i \) of the block where this operator acts on (we leave \( J_i \) undefined if this operator is not invoked).

Now for a block \( i \), we define the (random) set of iterations where WFP modifies the \( i \)-th block iterate \( \text{proj}_i \tilde{z}^\tau \): \( J_i = \{ t \geq 1 : \text{proj}_i \tilde{z}^t \neq \text{proj}_i \tilde{z}^{t-1} \} \).

Consider a block \( i \). Now we claim that the sequence \( (\text{proj}_i \tilde{z}^t)_{t \in J_i \cup \{0\}} \) has the same distribution as the sequence of binary iterates obtained by applying WFP to the block \( P_i \) alone. More precisely, as we defined \( \tilde{z}^t \), let \( \tilde{w}^t \in \{0,1\}^{n_i} \) be the iterate at the end of iteration \( t \) when we apply WFP to the block \( P_i \) with starting point \( \tilde{w}^0 = \text{proj}_i \tilde{x}^0 \) (notice we used the letter \( w \) to replace the letter \( x \) used in the description of WFP). To avoid ambiguity, we use WFP\(_P\) to refer to the execution of the algorithm over \( P \) and and WFP\(_{P_i}\) to refer to the execution of the algorithm over \( P_i \).

Lemma 4.14. The sequences \( (\text{proj}_i \tilde{z}^t)_{t \in J_i \cup \{0\}} \) and \( (\tilde{w}^t)_{t \geq 0} \) have the same distribution.

Proof. Before we start, notice that each iteration of algorithm WFP over \( P \) is either an application of AltProj\(_P\) or an application of AltProj\(_{P_i}\) followed by \( \text{RandWalkSAT}_i \) (the subscript \( P \) in AltProj\(_P\) makes explicit to which set the \( l_1 \)-projection is happening). Moreover, because of the decomposability of the instance, the operator AltProj commutes with the projection \( \text{proj}_i \):
\[
\text{proj}_i \circ \text{AltProj}_P = \text{AltProj}_{P_i} \circ \text{proj}_i.
\]

Now we start comparing the sequences \( (\text{proj}_i \tilde{z}^t)_{t \in J_i \cup \{0\}} \) and \( (\tilde{w}^t)_{t \geq 0} \) using a coupling argument. The idea is to show that if at some point both sequences have the same iterate, then the next item of both
sequences have the same distribution, which can then be coupled to continue this process (see [Tho00] for a formal presentation of this coupling argument).

We proceed by induction. By definition both sequences have the same starting point. Now consider the \( i \)th smallest index in \( I_i \), denoted by \( t_i \), and assume by induction that \( \text{proj}_i \) and \( \tilde{\omega}^i \) have the same distribution; we couple them so as to have \( \text{proj}_i \tilde{x}^j = \tilde{\omega}^j \).

If \( \text{proj}_i(\tilde{x}^j) \) belongs to \( \text{proj}_{\text{bin}} P_i \), then WFP will not change this part of the iterate anymore and \text{RandWalkSAT} is not invoked in the \( i \)th block anymore (since the constraint in the \( i \)th block is satisfied it cannot be used in the minimal certificate). In this case, \( t_i \) is the last index in \( I_i \), i.e., \( \tilde{x}^j \) is the last item of the sequence of iterates. Since \( \text{proj}_i \tilde{x}^j = \tilde{\omega}^j \), the same holds for \( \tilde{\omega}^j \). Thus, there is no inductive step to be proved in this case.

Now suppose \( \text{proj}_i(\tilde{x}^j) \notin \text{proj}_{\text{bin}} P_i \). We have two cases.

**Case 1: AltProj**

**Case 1: AltProj** \( \text{AltProj}_P(\text{proj}_i(\tilde{x}^j)) \neq \text{proj}_i \tilde{x}^j \). Then notice \( \text{AltProj}_P(\tilde{x}^j) \neq \tilde{x}^j \), since

\[
\text{proj}_i \text{AltProj}_P(\tilde{x}^j) = \text{AltProj}_P(\text{proj}_i(\tilde{x}^j)) \neq \text{proj}_i \tilde{x}^j.
\]

Thus, WFP changes the iterate in the iteration \( t_j + 1 \), and hence the next index in \( I_i \) is \( t_{j+1} = t_j + 1 \). Moreover, because of this change, the operator \text{RandWalkSAT} is not invoked in this iteration of WFP and thus \( \tilde{x}^{j+1} = \text{AltProj}_P(\tilde{x}^j) \), which implies that in the \( i \)th block \( \text{proj}_i \tilde{x}^{j+1} = \text{AltProj}_P(\text{proj}_i \tilde{x}^j) \). The same observations hold for WFP on, so

\[
\tilde{\omega}^{j+1} = \text{AltProj}_P \tilde{\omega}^j = \text{AltProj}_P(\text{proj}_i \tilde{x}^j) = \text{proj}_i \tilde{x}^{j+1},
\]

proving the inductive step in this case.

**Case 2: AltProj**

**Case 2: AltProj** \( \text{AltProj}_P(\text{proj}_i(\tilde{x}^j)) = \text{proj}_i \tilde{x}^j \). Because of this fixed-point property, the iterate \( \tilde{x}^j \) remains the same for \( \tau \in \{t_j, \ldots, t_{j+1} - 1\} \). Moreover, since \( t_{j+1} \in I_i \), we have the iterate \( \tilde{x}^{j+1} \) different from \( \tilde{x}^j \); again because of the fixed-point property, it implies that at iteration \( t_{j+1} \) the algorithm WFP invokes \text{RandWalkSAT} on block \( i \). Thus, the iterate \( \text{proj}_i \tilde{x}^{j+1} \) is obtained by applying \text{RandWalkSAT} to \( \text{proj}_i \tilde{x}^j \) with the constraint of \( P_i \) as minimal projected certificate. For the same reason, algorithm WFP obtains \( \tilde{\omega}^{j+1} \) by applying \text{RandWalkSAT} to \( \tilde{\omega}^j \) with the constraint of \( P_i \) as minimal projected certificate. Since the initial points \( \text{proj}_i \tilde{x}^j = \tilde{\omega}^j \) are the same, it follows that \( \text{proj}_i \tilde{x}^{j+1} \) and \( \tilde{\omega}^{j+1} \) have the same distribution. This concludes the inductive step in this case, and thus the proof.

Using Theorem 4.4, with probability at least \( 1 - \frac{\delta}{k} \) algorithm WFP performs at most \( n_1(n_1 + 1) \cdot 2^{2n_1 \log n_1} \cdot \lceil \ln(k/\delta) \rceil \) iterations, and hence by the equivalence from the above lemma this provides an upper bound on the length of the sequence \( \text{proj}_j \tilde{x}^j \) for \( j \in \cup_{i \subset [0]} I_i \), or equivalently on the size of \( I_i \). Employing a union bound, with probability at least \( 1 - \delta \) we have that

\[
\sum_{i=1}^{k} |I_i| \leq \lceil \ln(k/\delta) \rceil \sum_{i=1}^{k} n_i(n_i + 1) \cdot 2^{2n_i \log n_i}.
\]

Since every iteration of algorithm WFP is accounted for in one of the sets \( I_i \), this upper bounds the number of iterations of the algorithm. This concludes the proof of Theorem 2.4.

5 Computations

In this section, we describe the algorithms that we have implemented and report computational experiments comparing the performance of the original Feasibility Pump 2.0 algorithm from [FS09], which we denote by FP, to our modified code that uses the new perturbation procedure. The code is based on the current version of the Feasibility Pump 2.0 code (the one available on the NEOS servers), which is implemented in C++ and linked to IBM ILOG CPLEX 12.6.3 [ILO] for preprocessing and solving LPs. All features such as constraint propagation which are part of the Feasibility Pump 2.0 code have been left unchanged.

All algorithms have been run on a cluster of identical machines, each equipped with an Intel Xeon CPU E5-2623 V3 running at 3.0GHz and 16 GB of RAM. Each run had a time limit of half an hour.
5.1 WalkSAT-based perturbation

In preliminary tests, we implemented the algorithm WFP (with \( \ell = 4 \)) as described in the previous section. However, its performance was not competitive with FP. In hindsight, this can be justified by the following reasons:

- Picking a fixed \( \ell \) can be tricky. Too small or too big a value can lead to slow convergence in practical implementations.
- Using RandWalkSAT\(_\ell\) at each perturbation step can be overkill, as in most cases the original perturbation scheme does just fine.
- Computing the minimal certificate can be too expensive, as it requires solving LPs.

For the reasons above, we devised a more conservative implementation of a perturbation procedure inspired by WalkSAT, which we denote by WFPb. The algorithm works as follows. Let \( F \subset [n] \) be the set of indices with positive fractionality \( |\tilde{x}_j - \bar{x}_j| \). If \( TT \leq |F| \), then the perturbation procedure is just the original one in FP. Else, let \( S \) be the union of the supports of the constraints that are not satisfied by the current point \((\tilde{x}, \bar{y})\). We select the \( |F| \) indices with largest fractionality \( |\tilde{x}_j - \bar{x}_j| \) and select uniformly at random \( \min\{ |S|, TT - |F| \} \) indices from \( S \), and flip the values in \( \tilde{x} \) for all the selected indices.

Note also that the above procedure applies only to the case in which a cycle of length one is detected. In case of longer cycle, we use the very same restart strategy of FP.

5.2 Computational results

We tested the three algorithms (FP, WFP, and WFPb) on two classes of models: two-stage stochastic models, and the MIPLIB 2010 dataset.

Two-stage stochastic models. In order to validate the hypothesis suggested by the theoretical results that our walkSAT-based perturbation should work well on almost-decomposable models, we compared the algorithms on two-stage stochastic models. These are the deterministic equivalent of two-stage stochastic programs and have the form

\[
\begin{align*}
Ax + D^i y^i &\leq c^i, \quad i \in \{1, \ldots, k\} \\
x &\in \{0,1\}^p \\
y^i &\in \{0,1\}^q, \quad i \in \{1, \ldots, k\}.
\end{align*}
\]

The variables \( x \) are the first-stage variables, and \( y^i \) are the second-stage variables for the \( i \)-th scenario. Notice that these second-stage variables are different for each scenario, and are only coupled through the first-stage variables \( x \). Thus, as long as the number of scenarios is reasonably large compared to dimensions of \( x, y^1, \ldots, y^k \), these problems are to some extent almost-decomposable.

For our experiments we randomly generated instances of this form as follows: (1) the entries in \( A \) and the \( D^i \)'s are independently and uniformly sampled from \{\(-10, \ldots, 10\}\}; (2) to guarantee feasibility, a 0/1 point is sampled uniformly at random from \{0,1\}^{p+kq} and the right-hand sides \( c^i \) are set to be the smallest ones that make this point feasible. We generated 150 instances, 15 for each setting of parameters \( k \in \{10, 20, 30, 40, 50\} \) and \( p \in \{10, 20\} \) (\( q \) is always set equal to \( p \)).

We compared the three algorithms over these instances using ten different random seeds. First, we aggregated the results based on the value of \( k \). The results are reported in Table 1. In the tables, #found denotes the number of models for which a feasible solution was found, while time and itr. report the shifted geometric means [Ach07] of running times and iterations (with shifts of 1s and 10 iterations), respectively. Column pgap reports the average primal gap of solutions found w.r.t. the best known solutions. For WFPb, we also report in column \( \text{wpertQ} \) the average percentage of WalkSAT-based perturbations.

Then we aggregated the results based on seed. The corresponding results are reported in Table 2, were the last row provides average figures across seeds.
On this testbed of models, both WFP and WFPb outperform FP; the first does so only marginally, while WFPb significantly improves over FP and across all performance measures. Notice that being a pure-integer testbed, WFP is not slowed down by the need of solving LPs to compute minimal certificates: still, the strategy of always using the WalkSAT-based perturbation is too aggressive and does not pay off as nicely as the strategy in WFPb. The current results do not show a different relationship between number of iterations and \( k \) among the different methods, as could be indicated from our theoretical findings. However, this is not surprising, as all methods either find a feasible solution or hit the time limit well before the theoretical worst case limits.

**MIPLIB 2010.** We also compared the algorithms on the whole MIPLIB 2010 [KAA+11], a testbed of 358 models. Again we compared the three algorithms using ten different random seeds. A seed by seed comparison is reported in Table 3.

The improvement in this heterogeneous testbed is less dramatic than in the two-stage stochastic models. In this case, WFP performs consistently worst than FP, according to all measures. On the other hand, WFPb still consistently dominates FP, albeit by a very small margin: it can find more solutions in 8 out 10 cases (in the remaining 2 cases it is a tie), taking a comparable number of iterations and computing time. Solution quality, as measured by the average primal gap, is also not negatively affected by the proposed change.

Finally, we also recomputed aggregated results filtering out all instances on which all methods could find a feasible solution in less than 10 iterations. A seed by seed comparison on this restricted testbed of harder instances is reported in Table 4. The results therein are consistent with those of the complete testbed.

In conclusion, given that the suggested modification is very simple to implement, and appears to dominate FP consistently, it suggests it is a good idea to add it as a feature in all future feasibility pump codes.

**Acknowledgments**

We would like to thank Andrea Lodi for discussions and clarifications on Feasibility Pump. We would also like to thank the anonymous reviewers for their invaluable comments and suggestions.
Table 3: Aggregated results by seed on MIPLIB2010.

<table>
<thead>
<tr>
<th>seed</th>
<th># found</th>
<th>itr.</th>
<th>#</th>
<th>time (s)</th>
<th>pgap</th>
<th>wperfQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FP</td>
<td>WFP</td>
<td>WFP</td>
<td>FP</td>
<td>WFP</td>
<td>WFP</td>
</tr>
<tr>
<td>1</td>
<td>279</td>
<td>282</td>
<td>272</td>
<td>43</td>
<td>55</td>
<td>8.24</td>
</tr>
<tr>
<td>2</td>
<td>279</td>
<td>275</td>
<td>254</td>
<td>44</td>
<td>56</td>
<td>8.40</td>
</tr>
<tr>
<td>3</td>
<td>277</td>
<td>280</td>
<td>270</td>
<td>41</td>
<td>55</td>
<td>8.32</td>
</tr>
<tr>
<td>4</td>
<td>280</td>
<td>282</td>
<td>271</td>
<td>42</td>
<td>53</td>
<td>8.07</td>
</tr>
<tr>
<td>5</td>
<td>276</td>
<td>277</td>
<td>271</td>
<td>42</td>
<td>54</td>
<td>8.26</td>
</tr>
<tr>
<td>6</td>
<td>277</td>
<td>278</td>
<td>270</td>
<td>42</td>
<td>55</td>
<td>8.29</td>
</tr>
<tr>
<td>7</td>
<td>278</td>
<td>281</td>
<td>274</td>
<td>43</td>
<td>51</td>
<td>8.17</td>
</tr>
<tr>
<td>8</td>
<td>273</td>
<td>277</td>
<td>269</td>
<td>43</td>
<td>52</td>
<td>8.16</td>
</tr>
<tr>
<td>9</td>
<td>279</td>
<td>275</td>
<td>271</td>
<td>42</td>
<td>52</td>
<td>8.13</td>
</tr>
<tr>
<td>10</td>
<td>278</td>
<td>281</td>
<td>274</td>
<td>43</td>
<td>53</td>
<td>8.33</td>
</tr>
<tr>
<td>Avg.</td>
<td>278</td>
<td>280</td>
<td>272</td>
<td>43</td>
<td>54</td>
<td>8.24</td>
</tr>
</tbody>
</table>

Table 4: Aggregated results by seed on hard models from MIPLIB2010.

<table>
<thead>
<tr>
<th>seed</th>
<th># found</th>
<th>itr.</th>
<th>#</th>
<th>time (s)</th>
<th>pgap</th>
<th>wperfQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FP</td>
<td>WFP</td>
<td>WFP</td>
<td>FP</td>
<td>WFP</td>
<td>WFP</td>
</tr>
<tr>
<td>1</td>
<td>119</td>
<td>120</td>
<td>112</td>
<td>191</td>
<td>281</td>
<td>17.76</td>
</tr>
<tr>
<td>2</td>
<td>119</td>
<td>119</td>
<td>115</td>
<td>202</td>
<td>286</td>
<td>18.34</td>
</tr>
<tr>
<td>3</td>
<td>118</td>
<td>125</td>
<td>110</td>
<td>194</td>
<td>280</td>
<td>17.93</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>122</td>
<td>111</td>
<td>185</td>
<td>268</td>
<td>17.41</td>
</tr>
<tr>
<td>5</td>
<td>116</td>
<td>117</td>
<td>111</td>
<td>188</td>
<td>276</td>
<td>17.09</td>
</tr>
<tr>
<td>6</td>
<td>117</td>
<td>118</td>
<td>110</td>
<td>190</td>
<td>286</td>
<td>17.85</td>
</tr>
<tr>
<td>7</td>
<td>118</td>
<td>121</td>
<td>114</td>
<td>191</td>
<td>263</td>
<td>17.50</td>
</tr>
<tr>
<td>8</td>
<td>113</td>
<td>117</td>
<td>109</td>
<td>196</td>
<td>261</td>
<td>17.75</td>
</tr>
<tr>
<td>9</td>
<td>122</td>
<td>122</td>
<td>115</td>
<td>189</td>
<td>255</td>
<td>17.23</td>
</tr>
<tr>
<td>10</td>
<td>118</td>
<td>122</td>
<td>115</td>
<td>189</td>
<td>265</td>
<td>17.36</td>
</tr>
<tr>
<td>Avg.</td>
<td>118</td>
<td>120</td>
<td>112</td>
<td>191</td>
<td>281</td>
<td>17.62</td>
</tr>
</tbody>
</table>

Santanu S. Dey and Andres Iroume would like to gratefully acknowledge the support of NSF grants CMMI 1562578 and CMMI 1149400 respectively.

References


Appendix

A Original Feasibility Pump stalls even when flipping variables with zero fractionality is allowed

In Section 2 we showed that the original Feasibility Pump without restarts may stall; we now show that this is still the case even if variables with zero fractionality can be flipped in the perturbation step.

Let $\mathbf{T}$, the number of variables to be flipped, be randomly selected from the set $[t, T] \cap \mathbb{Z}$, where $T \in \mathbb{Z}_{++}$ is a pre-determined constant in the FP code (independent of the instance). Moreover assume the reasonable convention that for two variables with equal fractionality, we break ties using their index number, that is, if the $x_i$ and $x_j$ have the same fractionality and $i < j$, then $x_i$ is picked before $x_j$ to be flipped.

Consider the following subset-sum problem:

$$\max \quad x_{T+2}$$

s.t.  

$$5x_1 + \cdots + 5x_{T+1} + 2x_{T+2} = 5T + 5$$

$$x_i \in \{0, 1\} \forall i \in [T+2]$$

Clearly the LP optimal solution $\bar{x}^0$ is of the form $\bar{x}_{T+2}^0 = 1$, $\bar{x}_i^0 = \frac{5}{T}$ for some $i \in [T+1]$ and $\bar{x}_j^0 = 1$ for all $j \in [T+1] \setminus \{i\}$. Rounding this we obtain $\tilde{x}^0$ that is the all $1$’s vector. It is also straightforward to verify that $\bar{x}^0$ is a stalling solution (i.e., AltProj($\tilde{x}^0$) = $\tilde{x}^0$). So that algorithm randomly selects $\mathbf{T}$ from the set $[t, T] \cap \mathbb{Z}$ and flips $\mathbf{T}$ variables. Note that only one variable $x_i$ (with $i \in [T+1]$) has fractionality $\frac{1}{T} - 1$ and all other variables have fractionality $0$. So using the convention for breaking ties, we flip $x_i$ and $\mathbf{T} - 1$ other variables. Let $S$ be the set of flipped variables and, since $\mathbf{T} - 1 < T + 1$, the variable $x_{T+2}$ is not flipped. Thus, the point $\tilde{x}$ obtained after slipping has $\tilde{x}_{T+2} = 1$, $\tilde{x}_j = 0$ for $j \in S$, and $\tilde{x}_j = 1$ for $j \in [T+1] \setminus S$.

First note that $\tilde{x}$ is not a feasible solution since $\tilde{x}_{T+2} = 1$. Moreover,

1. If $S = \emptyset$, then $\tilde{x}$ is again the stalling point $\tilde{x}^0$.
2. If $S \neq \emptyset$, then $5\tilde{x}_1 + \cdots + 5\tilde{x}_{T+1} + 2\tilde{x}_{T+2} < 5T + 5$ and after projecting to the LP relaxation we obtain a point of the form of $\tilde{x}^0$ (i.e. exactly one of the coordinates $i \in [T+1]$ equals $\frac{5}{T}$, all others equal 1). Rounding this solution again gives us the stalling point $\tilde{x}^0$.

Thus, the algorithm simply keeps revisiting the same $0/1$ point $\tilde{x}^0$. This completes the proof of the claim.

B Proof of Theorem 4.3

**Lemma B.1.** Suppose that the following is a sequence of points visited by Feasibility Pump (without any randomization):

$$(\tilde{x}^1, \tilde{y}^1) \to (\tilde{x}^1, \tilde{y}^1) \to (\tilde{x}^2, \tilde{y}^2) \to (\tilde{x}^2, \tilde{y}^2),$$

where $(\tilde{x}^i, \tilde{y}^i)$, $i \in \{1, 2\}$ are the vertices of the LP relaxation $P$, $\tilde{x}^i$, $i \in \{1, 2\}$ are 0/1 vectors, $\tilde{x}^i = \text{round}(\tilde{x}^i)$ and $(\tilde{x}^2, \tilde{y}^2) = \ell_1\text{-proj}(P, \tilde{x}^1)$. Then,

$$\| \tilde{x}^1 - \tilde{x}^1 \|_1 \geq \| \tilde{x}^2 - \tilde{x}^1 \|_1 \geq \| \tilde{x}^2 - \tilde{x}^2 \|_1.$$

**Proof.** This result holds due to the fact that we are sequentially projecting using the same norm. In particular, we have that

$$\| \tilde{x}^1 - \tilde{x}^1 \|_1 \geq \| \tilde{x}^2 - \tilde{x}^1 \|_1,$$

since $(\tilde{x}^2, \tilde{y}^2) = \ell_1\text{-proj}(P, \tilde{x}^1)$, i.e. $\tilde{x}^2$ is a closest point in $\ell_1$-norm to $\tilde{x}^1$ in the projection of the LP relaxation in the $x$-space. Then

$$\| \tilde{x}^2 - \tilde{x}^1 \|_1 \geq \| \tilde{x}^2 - \tilde{x}^2 \|_1,$$

since $\tilde{x}^1$ and $\tilde{x}^2$ are both integer points and $\tilde{x}^2$ is obtained by rounding $\tilde{x}^2$ (and a rounded point is a closest integer point in $\ell_1$ norm).  

20
A long cycle in feasibility pump is a sequence
\[(\bar{x}^1, \bar{y}^1) \to (\bar{x}^1, \bar{y}^1) \to (\bar{x}^2, \bar{y}^2) \to (\bar{x}^2, \bar{y}^2) \to \ldots (\bar{x}^k, \bar{y}^k) \to (\bar{x}^k, \bar{y}^k)\]
where
1. \((\bar{x}^i, \bar{y}^i), i \in \{1, 2, \ldots, k\}\) are the vertices of the LP relaxation \(P, \bar{x}^i, i \in \{1, 2, \ldots, k\}\) are \(0-1\) vectors, \(\bar{x}^i = \text{round}(\bar{x}^i)\) and \((\bar{x}^{i+1}, \bar{y}^{i+1}) = \ell_1\text{-proj}(P, \bar{x}^i)\),
2. \(\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^{k-1}\) are unique integer vectors,
3. \(\bar{x}^1 = \bar{x}^k, \bar{x}^1 = \bar{x}^k,\) and
4. \(k \geq 3\).

The statement of Theorem 4.3 is that such a scenario cannot occur, assuming 0.5 is always rounded consistently.

**Proof of Theorem 4.3.** Without loss of generally, we assume that 0.5 is rounded up to the value 1. By contradiction, consider a long cycle as described above. We claim that for all \(i\) we have the coordinate-wise domination \(\bar{x}^{i+1} \geq \bar{x}^i\), which contradicts this long cycle.

To show this domination, first notice that by Lemma B.1 the sequence of \(\ell_1\) gaps \(||\bar{x}^i - \bar{x}^i||_1\) is non-decreasing, and because of the cycle we have that the first and last term of this sequence is the same; thus, all these gaps are the same. Hence, Lemma B.1 becomes equality: we have
\[||\bar{x}^i - \bar{x}^i||_1 = ||\bar{x}^{i+1} - \bar{x}^i||_1 = ||\bar{x}^{i+1} - \bar{x}^{i+1}||_1\]
for all \(i\). Letting \(J\) denote the set of indices \(j\) where \(\bar{x}_j^i \neq \bar{x}_j^{i+1}\), we can expand the last displayed equality to get
\[
\sum_{j} |\bar{x}_j^i - \bar{x}_j^{i+1}| = \sum_{j \notin J} |\bar{x}_j^i - \bar{x}_j^{i+1}| + \sum_{j \in J} |\bar{x}_j^{i+1} - \bar{x}_j^{i+1}|
\]
\[
= \sum_{j \notin J} |\bar{x}_j^i - \bar{x}_j^{i+1}| = \sum_{j \in J} |\bar{x}_j^{i+1} - \bar{x}_j^{i+1}|. \quad (8)
\]

Consider an index \(j \in J\). If \(\bar{x}_j^i = 0\), and thus \(\bar{x}_j^{i+1} = 1\), we have that \(\bar{x}_j^{i+1} \geq 0.5\) (since \(\bar{x}_j^{i+1} = 1\) was obtained from it by rounding) and thus \(|\bar{x}_j^i - \bar{x}_j^{i+1}| \geq |\bar{x}_j^{i+1} - \bar{x}_j^{i+1}|\). Similarly, if \(\bar{x}_j^i = 1\), and thus \(\bar{x}_j^{i+1} = 0\), we have that \(\bar{x}_j^{i+1} < 0.5\) and thus the strict inequality \(|\bar{x}_j^i - \bar{x}_j^{i+1}| > |\bar{x}_j^{i+1} - \bar{x}_j^{i+1}|\). In order to have the equality in (8) we thus cannot have any index \(j \in J\) with \(\bar{x}_j^i = 1\) and \(\bar{x}_j^{i+1} = 0\). Therefore, we have the domination \(\bar{x}^{i+1} \geq \bar{x}^i\). This concludes the proof. \(\square\)