SIAM J. OPTIM.
Vol. 28, No. 1, pp. 355–378
© 2018 Society for Industrial and Applied Mathematics

IMPROVING THE RANDOMIZATION STEP IN FEASIBILITY PUMP*

SANTANU S. DEY†, ANDRES IROUME†, MARCO MOLINARO‡, AND DOMENICO SALVAGNIN§

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 the 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 a 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.

Key words. integer programming, primal heuristic, feasibility pump

AMS subject classification. 90C11

DOI. 10.1137/16M1095962

1. Introduction.

Primal heuristics are used within mixed-integer linear programming (MILP) solvers for finding good integer feasible solutions quickly [FL11]. *Feasibility Pump* is a very successful primal heuristic for mixed-binary linear programs (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: **Input:** mixed-binary LP (with binary variables x and continuous variables y)
2: Solve the linear programming relaxation, and let $(\bar{x}, \bar{y})$ be an optimal solution
3: **while** $\bar{x}$ is not integral **do**
4: (Round) Round each coordinate of $\bar{x}$ to the closest integer, call the obtained vector $\widetilde{x}$
5: (Project) Let $(\bar{x}, \bar{y})$ be the point in the LP relaxation that minimizes $\sum_i |x_i - \widetilde{x}_i|$
6: **end while**
7: Return $(\bar{x}, \bar{y})$

*Received by the editors September 30, 2016; accepted for publication (in revised form) October 18, 2017; published electronically February 6, 2018.
http://www.siam.org/journals/siopt/28-1/M109596.html
Funding: The work of the first and second author was supported by NSF grants CMMI 1562578 and CMMI 1149400, respectively. The work of the third author was supported by the grant CNPq Universal 431480/2016-8. The work of the fourth author was supported by Miur Italy, project PRIN 2015 "Nonlinear and Combinatorial Aspects of Complex Networks."
†School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332 (santanu.dey@isye.gatech.edu, airoume3@gatech.edu).
‡Computer Science Department, PUC-Rio, Rio de Janeiro, Brazil (mmolinaro@inf.puc-rio.br).
§IBM Italy and DEI, University of Padova, Padova, Italy (dominiqs@gmail.com).

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 letting N be the number of variables with positive fractionality, randomly generate a positive integer $\mathbf{TT}$ and flip $\min\{\mathbf{TT}, N\}$ variables with largest fractionality.

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

Due to its success, many improvements and generalizations of Feasibility Pump, both for MILPs and mixed integer nonlinear programs (MINLPs), have been studied [AB07, BFL07, BCLM09, FS09, SLR13, DFLL10, BEET12, DFLL12, BEE+14]. 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 approaches avoid the randomization step altogether; see [BCLM09] for convex MINLPs and [GMSS17] for MILPs.

Moreover, even though Feasibility Pump 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 are no known bounds on the expected running-time of Feasibility Pump.

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 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 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 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, RandWalkSAT$_\ell$ constructs a *minimal (projected) infeasibility certificate* for this solution and *randomly picks* an ℓ 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 RandWalkSAT$_\ell$.

First, we analyze what happens if we simply repeatedly use *only* the new proposed randomization step RandWalkSAT$_\ell$, which gives a simple primal heuristic that we denote by mbWalkSAT (where "mb" indicates it is an extension of WalkSAT for *mixed-binary* problems). Not only do we show that 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:

$$P^I = P_1^I \times \cdots \times P_k^I, \text{ where for all } i \in [k] \text{ we have}$$

$$P_i^I = P_i \cap (\{0,1\}^{n_i} \times \mathbb{R}^{d_i}),\ P_i = \{(x^i, y^i) \in [0,1]^{n_i} \times \mathbb{R}^{d_i} : A^i x^i + B^i y^i \le c^i\}. \tag{1}$$

$$\text{Let } P = P_1 \times \cdots \times P_k \text{ denote the LP relaxation of } P^I.$$

Note that since we allow $k = 1$, this also captures a general mixed-binary set. We then have the following running-time guarantee for the primal heuristic MBWALKSAT.

THEOREM 2.1. *Consider a feasible decomposable mixed-binary set as in* (1). *Let* s_i *be such that each constraint in* P_i^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* $1 - \delta$, 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* $\bar{n}k\, 2^{\bar{n} \log \bar{n}} \cdot \ln(k/\delta)$, *where* $\bar{n} = \max_i n_i$.

There are a few interesting features of this bound that indicate 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* 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$_\ell$ *in conjunction with Feasibility Pump.* Next, we analyze RANDWALKSAT$_\ell$ 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 in 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:*

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

In particular, this class of instances includes *subset-sum* instances (i.e., $\{x \in \{0,1\}^n : ax = c\}$ with nonnegative a, c) and *knapsacks* in standard form (i.e., $\{(x, y) \in \{0,1\}^n \times \mathbb{R}_+ : ax + by = c\}$ with a, c nonnegative 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 nonzero probability [BFL07]) may not even converge, as illustrated next.

Remark 2.3. Consider the feasible subset-sum problem

$$\begin{aligned} \max\ & x_2 \\ s.t.\ & 3x_1 + x_2 = 3, \\ & x_1, x_2 \in \{0,1\}. \end{aligned}$$

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 $(\frac{2}{3}, 1)$. This solution is then rounded to the point $(1, 1)$, which is infeasible. This point is then ℓ_1-projected to the LP, giving back the point $(\frac{2}{3}, 1)$, 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 $|\frac{2}{3} - 1| = \frac{1}{3}$, 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 ℓ_1-projected to the LP, giving again the point $(\frac{2}{3}, 1)$. 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 Appendix A 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*

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

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$_\ell$, 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 in 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 nonzero probability of generating an iterate closer to a *coordinatewise maximal* feasible solution (Lemma 4.12), so the algorithm has some chance of "unstalling" to a point closer to a feasible solution.

Computational experiments. While the analyses above give insights on the usefulness of using RANDWALKSAT$_\ell$ in the randomization step of Feasibility Pump, 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$_\ell$. 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: In section 3 we discuss and present our analysis of the proposed randomization scheme RandWalkSAT$_\ell$; section 4 presents the analysis of the new randomization scheme RandWalkSAT$_\ell$ in conjunction with feasibility pump, and section 5 describes details of our empirical experiments.

Notation. We use $\mathbb{R}_+$ to denote the nonnegative 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 $\|v\|_1 = \sum_i |v_i|$ to denote the ℓ_1 norm. Finally, we use $e^i \in \mathbb{R}^n$ to denote the ith canonical basis vector.

3. New randomization step RandWalkSAT$_\ell$.

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$_\ell$, 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 selects 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 $(2(1 - \frac{1}{s}))^n$, 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 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 $\text{proj}_{bin}\, P$ 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 $\bar{x} \notin \text{proj}_{bin}\, P$, 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 λ 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, Carathéodory'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 $\bar{x} \notin \text{proj}_{bin} P$. 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 RANDWALKSAT$_\ell$ (notice that the condition $\lambda B = 0$ guarantees that a projected certificate has the form $\pi x \leq \pi_0$).

Algorithm 2 RANDWALKSAT$_\ell(\bar{x})$.

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

Note that in the pure-binary case and $\ell = 1$, this reduces to the main step executed during WALKSAT. We remark that the flexibility of introducing the parameter ℓ will be needed in section 4.

3.2. Analyzing the behavior of RandWalkSAT$_\ell$. In this section we consider the behavior of the algorithm MBWALKSAT that tries to find a feasible mixed-binary solution by just repeatedly applying the randomization step RANDWALKSAT$_\ell$.

Algorithm 3 MBWALKSAT.

1: **input parameter:** Integer $\ell \geq 1$
2: (Starting solution) Consider any mixed-binary point $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \{0,1\}^n \times \mathbb{R}^d$
3: **loop**
4: **if** $\bar{\mathbf{x}}$ does not belong to $\text{proj}_{bin} P$ **then**
5: RANDWALKSAT$_\ell(\bar{\mathbf{x}})$
6: **else**
7: (Output feasible lift of $\bar{\mathbf{x}}$) Find $\bar{\mathbf{y}} \in \mathbb{R}^d$ such that $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in P$, return $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$
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 (1), and let certSupp_i denote the maximum support size of a minimal projected certificate for the instance P^I_i which consists only of the ith block.

THEOREM 3.3 (Theorem 2.1 restated). *Consider a feasible decomposable mixed-binary set as in (1). Then with probability at least $1-\delta$,* MBWALKSAT *with parameter $\ell = 1$ returns a feasible solution within $T = \lceil \ln(k/\delta) \rceil \sum_i n_i \, 2^{n_i \log certSupp_i}$ iterations.*

In light of Lemma 3.2, if each constraint in P_i has at most s_i integer variables, we have $\text{certSupp}_i \leq \min\{s_i \cdot (d_i + 1), n_i\}$, and thus this statement indeed implies

Theorem 2.1 stated in the introduction. We remark that similar guarantees can be obtained for general ℓ, 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^I, then with high probability the algorithm returns a feasible solution within $n_i \ 2^{n_i \log \text{certSupp}_i} \cdot \ln(1/\delta)$ iterations. This analysis is inspired by the one given by Schöning [Sch99] and argues that with a small, but nonzero, probability the iteration of the algorithm makes the iterate $\bar{\boldsymbol{x}}$ 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^I; 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 (2). Given such a mixed-binary set P^I, we use certSupp = 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* (2)*. The probability that* MBWALKSAT *does not find a feasible solution within the first T iterations is at most $(1-p)^{\lfloor T/n \rfloor}$, where $p = certSupp^{-n}$. In particular, for $T = n \cdot 2^{n \log(certSupp)} \cdot \lceil \ln(1/\delta) \rceil$ this probability is at most δ (this follows from the inequality $(1-x) \le e^{-x}$ valid for $x \ge 0$).*

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

Fix an iteration t. To understand the probability that $\boldsymbol{X}_t = 0$, suppose that in this iteration $\bar{\mathbf{x}}$ does not belong to $\text{proj}_{bin} P$, and let $\pi x \le \pi_0$ be the minimal projected certificate for it used in RANDWALKSAT$_1$. Since the feasible point x^* satisfies the inequality $\pi x \le \pi_0$ but $\bar{\mathbf{x}}$ does not, there must be at least one index $\mathbf{i}^*$ in the support of π such where x^* and $\bar{\mathbf{x}}$ differ. Then if algorithm MBWALKSAT makes a "lucky move" and chooses $\mathbf{I} = \{\mathbf{i}^*\}$ 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, $\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - 1$. Moreover, since $\mathbf{I}$ is independent of $\mathbf{i}$, the probability of choosing $\mathbf{I} = \{\mathbf{i}^*\}$ is $1/|\text{supp}(\pi)| \ge 1/\text{certSupp}$.

Therefore, if we start at iteration t and for all the next $\boldsymbol{X}_t$ iterations either the iterate belongs to $\text{proj}_{bin} P$ or the algorithm makes a lucky move, it terminates by time $t + \boldsymbol{X}_t$. Thus, with probability at least $(1/\text{certSupp})^{\boldsymbol{X}_t} \ge (1/\text{certSupp})^n = p$ the algorithm terminates by time $t + \boldsymbol{X}_t \le t + n$.

To conclude the proof, let $\alpha = \lfloor T/n \rfloor$ and call iterations $i \cdot n, \ldots, (i+1) \cdot n - 1$, the ith 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 ith block. Putting these bounds together for all α blocks, the probability that the algorithm *does not* stop by the end of block α is at most $(1-p)^\alpha$. This concludes the proof. ∎

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 (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* (1). *Consider a point* $\bar{x} \notin \text{proj}_{bin}\, P$ *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,* $\bar{x}^j \notin \text{proj}_{bin}\, P_j$.

Proof. Let us use the natural decomposition $\bar{x} = (\bar{x}^1, \ldots, \bar{x}^k) \in \{0,1\}^{n_1} \times \cdots \times \{0,1\}^{n_k}$, and call the certificate $(\pi x \leq \pi_0) \triangleq (\sum_i \lambda^i A^i x^i \leq \sum_i \lambda^i b^i)$. By the definition of a 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 $\lambda^i B^i = 0$ for all i, so, in particular, for j. Thus, the inequality $\lambda^j (A^j, B^j)(x^j, y^j) \leq \lambda^j b^j$ 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.

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

We can finally prove the desired theorem.

Proof of Theorem 3.3. Again we use the natural decomposition $\bar{\mathbf{x}} = (\bar{\mathbf{x}}^1, \ldots, \bar{\mathbf{x}}^k) \in \{0,1\}^{n_1} \times \cdots \times \{0,1\}^{n_k}$ 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_j^I's, namely, the P_j^I containing all the inequalities in the minimal projected certificate used in this iteration; let $\mathbf{J}_t \in [k]$ denote the (random) index j of the block associated with iteration t. Notice that at iteration t, only the binary variables $x^{\mathbf{J}_t}$ can be modified by the algorithm.

Let $T_i = n_i\, 2^{n_i \log n_i} \lceil \ln(k/\delta) \rceil$. Applying the proof of Theorem 3.4 to the iterations $\{t : \mathbf{J}_t = i\}$ with index i, we get that with probability at least $1 - \frac{\delta}{k}$ the algorithm finds some $\bar{\mathbf{x}}^i$ in $\text{proj}_{bin}\, P_i$ 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 $\mathbf{J}_t \neq i$, 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 $\bar{\mathbf{x}}^i \in \text{proj}_{bin}\, P_i$ 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 $\bar{\mathbf{x}}$ belongs to $\text{proj}_{bin}\, P$ 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 (2). Given a 0/1 point $\widetilde{x} \in \{0,1\}^n$, let ℓ_1-proj$(P, \widetilde{x})$ denote a point (x, y) in P where $\|\widetilde{x} - x\|_1$ is as small as possible. In case of ties, we define ℓ_1-proj to have the following property.

PROPERTY 4.1 (ℓ_1-projection gives vertex). *Consider a point* $\widetilde{x} \in \{0,1\}^n$ *not in* $\text{proj}_{bin}\, P$, *and let* $(\bar{x}, \bar{y}) = \ell_1$*-proj*$(P, \widetilde{x})$. *Then* $\bar{x}$ *is a vertex of* $\text{proj}_{bin}\, P$.

Indeed notice that since $\text{proj}_{bin}\, P \subseteq [0,1]^n$ and ℓ_1-proj$(P, \widetilde{x})$ is an LP problem whose objective function only depends on the x-components, there is always a vertex

x of $\text{proj}_{bin} P$ where (x, y) satisfies the desired properties of ℓ_1-proj$(P, \widetilde{x})$ 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 "ℓ_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 $\widetilde{x}^0 = \text{round}(\bar{x}^0)$
4: **for** t = 1,2,... **do**
5: $(\bar{\mathbf{x}}^t, \bar{\mathbf{y}}^t) = \ell_1\text{-proj}(P, \widetilde{\mathbf{x}}^{t-1})$
6: $\widetilde{\mathbf{x}}^t = \text{round}(\bar{\mathbf{x}}^t)$
7: **if** $\widetilde{\mathbf{x}}^t \in \text{proj}_{bin}(P)$ **then**
8: Return any $(\widetilde{\mathbf{x}}^t, \widetilde{\mathbf{y}}^t) \in P$
9: **end if**
10: **if** $\widetilde{\mathbf{x}}^t = \widetilde{\mathbf{x}}^{t-1}$ **then** ▷ iterations have stalled
11: $\widetilde{\mathbf{x}}^t = \text{RANDWALKSAT}_\ell(\widetilde{\mathbf{x}}^t)$
12: **end if**
13: **end for**

(In step 7 it suffices to test whether $(\widetilde{\mathbf{x}}^t, \bar{\mathbf{y}}^t) \in P$ and return this point: This is because whenever we get $\widetilde{\mathbf{x}}^t \in \text{proj}_{bin}(P)$, in the next iteration the projection step will compute $(\bar{\mathbf{x}}^{t+1}, \bar{\mathbf{y}}^{t+1}) \in P$ with the same 0/1 part $\bar{\mathbf{x}}^{t+1} = \widetilde{\mathbf{x}}^t$, which stays the same after rounding, and thus $(\widetilde{\mathbf{x}}^{t+1}, \bar{\mathbf{y}}^{t+1}) \in P$. Also, since RANDWALKSAT$_\ell$ 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 $\widetilde{\mathbf{x}}^t = \widetilde{\mathbf{x}}^{t-1}$. In principle, there could be "long cycle" stalling, that is, $\widetilde{\mathbf{x}}^t = \widetilde{\mathbf{x}}^{t'}$, where $t' < t-1$ but $\widetilde{\mathbf{x}}^{t'}, \ldots, \widetilde{\mathbf{x}}^{t-1}$ 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 ℓ_1-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 $(\check{x}^1, \ldots, \check{x}^k) = \ell_1\text{-proj}(P, x)$, then $\check{x}^i = \ell_1\text{-proj}(P_i, x^i)$ 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 nonempty 1-row set* $P^I = P \cap (\{0,1\}^n \times \mathbb{R}^d_+)$ *for*

$$P = \{(x,y) \in [0,1]^n \times \mathbb{R}^d_+ : ax + by = c\}. \tag{3}$$

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* $(1-p)^{\lfloor T/(n\cdot(n+1))\rfloor}$*, where* $p = (1/n^2)^n$*. In particular, for* $T = n(n+1)\cdot 2^{2n\log n}\cdot\lceil\ln(1/\delta)\rceil$ *this probability is at most* δ*.*

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 $x^* \in \text{proj}_{bin}\, P$ and track its distance to the iterates $\widetilde{\boldsymbol{x}}^t$ generated by algorithm WFP. However, while again the randomization step RANDWALKSAT$_2$ brings $\widetilde{\boldsymbol{x}}^t$ closer to x^* with small but nonzero probability, the issue is that the projections ℓ_1-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 coordinatewise 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$_2$ *plus* "ℓ_1-proj + round" still has a nonzero 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 ℓ_1-proj. Given a point $\widetilde{x} \in \{0,1\}^n$, let AltProj($\widetilde{x}$) be the effect of applying to $\widetilde{x}$ the function ℓ_1-proj(.) and then round(.), namely, if $(\bar{x},\bar{y}) = \ell_1\text{-proj}(\widetilde{x})$ then AltProj($\widetilde{x}$) = round($\bar{x}$). Notice this is again a 0/1 vector; moreover, if $\widetilde{x}$ belongs to $\text{proj}_{bin}\, P$, then AltProj($\widetilde{x}$) = $\widetilde{x}$. Then algorithm WFP can be thought of as performing an AltProj operation, then checking if the iterate obtained either belongs to $\text{proj}_{bin}\, P$ (in which case it exits) or if it equals the previous iterate (in which case it applies RANDWALKSAT$_2$); 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 $\text{AltProj}^t(\widetilde{x}) = \text{AltProj}\left(\text{AltProj}^{t-1}(\widetilde{x})\right)$ (with $\text{AltProj}^1 = \text{AltProj}$), and if the sequence $(\text{AltProj}^t(\widetilde{x}))_t$ stabilizes at a point, let $\text{AltProj}^*(\widetilde{x})$ denote this point. We then arrive at the compressed version of the algorithm WFP.

Algorithm 5 WFP-Compressed.

input parameter: integer $\ell \geq 1$
Let $(\bar{x}^0, \bar{y}^0)$ be an optimal solution of the LP relaxation
Let $z^0 = \text{round}(\bar{x}^0)$
for τ = 1,2,... **do**
  $\widetilde{\boldsymbol{z}}^\tau = \text{AltProj}^*(\boldsymbol{z}^{\tau-1})$
  if $\widetilde{\boldsymbol{z}}^\tau \in \text{proj}_{bin}\, P$ **then**
    Return $\widetilde{\boldsymbol{z}}^\tau$
  end if
  $\boldsymbol{z}^\tau = \text{RANDWALKSAT}_\ell(\widetilde{\boldsymbol{z}}^\tau)$
end for

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

By using the same randomness in both WFP and WFP-compressed (or more precisely coupling the outcomes of RandWalkSAT$_\ell$(.) 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:

$$\begin{aligned}&\#\text{ iterations WFP} \le\\ &\quad[\#\text{ iterations WFP-Compressed}]\cdot \max_{\widetilde{x}\in\{0,1\}^n} \min\{k : \text{AltProj}^k(\widetilde{x}) = \text{AltProj}^*(\widetilde{x})\}.\end{aligned} \tag{4}$$

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 ℓ_1-projection it performs.

Lemma 4.5. *The following hold:*

1. *The set* $\text{proj}_{bin}\, P$ *is equal to either the set* $[0,1]^n$*, the set* $\{x \in [0,1]^n : ax \le c\}$*, the set* $\{x \in [0,1]^n : ax \ge c\}$*, or the set* $\{x \in [0,1]^n : ax = c\}$*.*
2. *For any point* $\widetilde{x} \in \{0,1\}^n$*,* ℓ_1*-proj*$(\widetilde{x})$ *has at most one fractional coordinate.*
3. *For any point* $\widetilde{x} \in \{0,1\}^n$ *not in* $\text{proj}_{bin}\, P$ *or, equivalently,* $\|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1 > 0$*, we have* $a \cdot \ell_1\text{-proj}(\widetilde{x}) = c$*.*

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

2. Recall ℓ_1-proj$(\widetilde{x})$ is a vertex of $\text{proj}_{bin}\, P$. But since $\text{proj}_{bin}\, P$ 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.

3. The intuition if $\widetilde{x} \notin \text{proj}_{bin}\, P$, then the ℓ_1-projection of this point to $\text{proj}_{bin}\, P$ lies on the boundary of $\text{proj}_{bin}\, P$, and thus should satisfy the equality $ax = c$.

More precisely, let $\bar{x} = \ell_1$-proj$(\widetilde{x})$. Recall the classification of $\text{proj}_{bin}\, P$ from item 1. If $\text{proj}_{bin}\, P = [0,1]^n$, then $\widetilde{x} \in \text{proj}_{bin}\, P$, and if $\text{proj}_{bin}\, P = \{x \in [0,1]^n : ax = c\}$, then clearly $a\bar{x} = c$. So assume $\text{proj}_{bin}\, P = \{x \in [0,1]^n : ax \le c\}$ (the case $ax \ge c$ is analogous). By contradiction suppose $\widetilde{x} \notin \text{proj}_{bin}\, P$, so $a\widetilde{x} > c$, but $a\bar{x} < c$. Then there is $\varepsilon > 0$ such that $\varepsilon\widetilde{x} + (1-\varepsilon)\bar{x}$ belongs to $\text{proj}_{bin}\, P$. But this point is closer in ℓ_1 to $\widetilde{x}$ than $\bar{x}$ is, contradicting the minimality of the latter. This concludes the proof. ∎

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

Lemma 4.6. *Consider a point* $\widetilde{x} \in \{0,1\}^n$*. Then:*

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

Proof. Consider a point $\widetilde{x} \in \{0,1\}^n$ with $\|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1 \in (0, \frac{1}{2}]$ (in the case $\|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1 = 0$, item 1 clearly holds). This implies that the vectors ℓ_1-proj$(\widetilde{x})$ and $\widetilde{x}$ differ exactly in the unique (by the lemma above) fractional coordinate of $\widetilde{x}$;

let j denote this coordinate. This implies that after rounding we have $\text{AltProj}(\widetilde{x}) = \text{round}(\text{proj}(\widetilde{x}))_i = \widetilde{x}_i$ for all $i \neq j$.

This also implies that $|\ell_1\text{-proj}(\widetilde{x})_j - \widetilde{x}_j| = \|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1$. If the right-hand side is strictly less than $\frac{1}{2}$, we have $\text{round}(\ell_1\text{-proj}(\widetilde{x}))_j = \widetilde{x}_j$, and thus $\text{round}(\ell_1\text{-proj}(\widetilde{x})) = \widetilde{x}$; this proves item 1 of the lemma. If instead we have $\|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1 = \frac{1}{2}$, then $\ell_1\text{-proj}(\widetilde{x})_j$ must be equal to $\frac{1}{2}$, which implies that $\text{round}(\ell_1\text{-proj}(\widetilde{x}))_j = 1$. Thus, $\text{AltProj}(\widetilde{x})_j \geq \widetilde{x}_j$, 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 AltProj we end up in one of the cases of the lemma above.

LEMMA 4.7. *Consider $\widetilde{x} \in \{0,1\}^n$ and let $\widetilde{x}' = \text{AltProj}(\widetilde{x})$. Then*

$$\|\ell_1\text{-proj}(\widetilde{x}') - \widetilde{x}'\|_1 \leq \frac{1}{2}.$$

Proof. Let $\bar{x} = \ell_1\text{-proj}(\widetilde{x})$ and recall $\widetilde{x}' = \text{round}(\bar{x})$. Since $\bar{x}$ has at most one fractional coordinate (Lemma 4.5), this is the only one that can be rounded (to the nearest integer) and hence $\|\bar{x} - \widetilde{x}'\|_1 \leq \frac{1}{2}$. Since $\ell_1\text{-proj}(\widetilde{x}')$ is a minimizer of $\min_{x \in \text{proj}_{bin}\, P} \|x - \widetilde{x}'\|_1$ and $\bar{x}$ is a feasible solution for this minimization problem, we have $\|\ell_1\text{-proj}(\widetilde{x}') - \widetilde{x}'\|_1 \leq \frac{1}{2}$. This concludes the proof. ∎

Since after the first application of AltProj we satisfy the conditions of Lemma 4.6, and since this lemma guarantees that each further application of AltProj either does not do anything or componentwise 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 AltProj.

COROLLARY 4.8. *For any point $\widetilde{x} \in \{0,1\}^n$4, $\text{AltProj}^*(\widetilde{x}) = \text{AltProj}^{n+1}(\widetilde{x})$.*

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 RANDWALKSAT$_2$ when combined with 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 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 $\widetilde{x} = (0,0,0,0)$ and the sequence of iterates $(\text{AltProj}^t(\widetilde{x}))_t$. In the first step we have two options for the projection $\ell_1\text{-proj}(\widetilde{x}) = (0,0,0,0)$ (due to the symmetry between x_1 and x_3), so for concreteness assume $\ell_1\text{-proj}(\widetilde{x}) = (\frac{1}{2},0,0,0)$; notice this falls on item 2 of Lemma 4.6. After rounding we then have $\text{AltProj}(\widetilde{x}) = (1,0,0,0)$. In the next step, when projecting $\text{AltProj}(\widetilde{x})$ we again have two options (due now to a different symmetry between x_1 and x_2), so for concreteness assume $\ell_1\text{-proj}(\text{AltProj}(\widetilde{x})) = (1,\frac{1}{2},0,0)$ (again item 2 of Lemma 4.6); thus, $\text{AltProj}^2(\widetilde{x}) = (1,1,0,0)$. Proceeding in this way, we may obtain $\text{AltProj}^3(\widetilde{x}) = (1,1,1,0)$, at which point the sequence finally stabilizes: $\text{AltProj}^4(\widetilde{x}) = \text{AltProj}^3(\widetilde{x})$, which then equals $\text{AltProj}^*(\widetilde{x})$.

The next lemma show 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 $\widetilde{x} \in \{0,1\}^n$ with $\|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1 \leq \frac{1}{2}$. Consider the points $\widetilde{x}' = \text{AltProj}(\widetilde{x})$ and $\widetilde{x}'' = \text{AltProj}^2(\widetilde{x})$. Suppose $\widetilde{x} \neq \widetilde{x}' \neq \widetilde{x}''$, and from Lemma 4.6.2 let $i_1, i_2 \in [n]$ be the indices such that $\text{supp}(\widetilde{x}') = \text{supp}(\widetilde{x}) \cup \{i_1\}$ and $\text{supp}(\widetilde{x}'') = \text{supp}(\widetilde{x}') \cup \{i_2\}$. Then $a_{i_1} = -a_{i_2}$.*

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

Lemma 4.7 guarantees that $\|\bar{x}' - \widetilde{x}'\|_1 \leq \frac{1}{2}$, and because $\widetilde{x}' \neq \widetilde{x}'' = \text{AltProj}(\widetilde{x}')$, Lemma 4.6.1 guarantees that actually $\|\bar{x}' - \widetilde{x}'\|_1 = \frac{1}{2}$. Thus, the same argument above holds with $\widetilde{x}'$ replacing $\widetilde{x}$ and gives that $\bar{x}' = \widetilde{x}' + \frac{1}{2}e^{i_2} = \widetilde{x} + e^{i_1} + \frac{1}{2}e^{i_2}$. In particular, $\bar{x} - \bar{x}' = -\frac{1}{2}(e^{i_1} + e^{i_2})$.

From Lemma 4.5.3 we have that the points $\bar{x}$ and $\bar{x}'$ satisfy $ax = c$. Thus, taking their differences and using the equality above we obtain

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

which implies $a_{i_1} = -a_{i_2}$. This concludes the proof. ∎

Finally we start bringing RANDWALKSAT$_2$ into the picture. The next lemma shows that given any point $\widetilde{x}$, there is a "lucky choice" in RANDWALKSAT$_2(\widetilde{x})$ 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 ℓ_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 $\text{proj}_{bin}\, P$, and a point $\widetilde{x} \in \{0,1\}^n$ not in $\text{proj}_{bin}\, P$. Suppose AltProj$(\widetilde{x}) = \widetilde{x}$. Then there is a point $\widetilde{x}' \in \{0,1\}^n$ satisfying the following:*

1. *(close to $\widetilde{x}$) $\|\widetilde{x}' - \widetilde{x}\|_0 \leq 2$;*
2. *(closer to x^*) $\|\widetilde{x}' - x^*\|_0 \leq \|\widetilde{x} - x^*\|_0 - 1$;*
3. *(projection control) $\|\ell_1\text{-proj}(\widetilde{x}') - \widetilde{x}'\|_1 \leq \frac{1}{2}$.*

Moreover, if we have the equality $\|\ell_1\text{-proj}(\widetilde{x}') - \widetilde{x}'\|_1 = \frac{1}{2}$ in item 3, then $\|\widetilde{x}' - x^*\|_0 \leq \|\widetilde{x} - x^*\|_0 - 2$.

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

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

Define $\chi = \widetilde{x} - x^*$; so identifying $\widetilde{x}$ and x^* with the corresponding sets they indicate, $\chi_i = 1$ if i belongs to $\widetilde{x}$ but not x^*, $\chi_i = -1$ if i belongs to x^* but not $\widetilde{x}$, and 0

otherwise. Thus, to construct an $\widetilde{x}'$ that is closer to x^* than $\widetilde{x}$, we will subtract from $\widetilde{x}$ one or two terms χ_i's.

Since

$$0 \prec a\widetilde{x} - c \prec a(\widetilde{x} - x^*) = \sum_i a_i \chi_i, \tag{5}$$

there is at least one index where $a_i\chi_i \succ 0$; we break into two cases depending on how $\succ$-big such a value can be.

Case 1: There is index j such that $\chi_j a_j \succ 0$ and $\chi_j a_j \preceq a\widetilde{x} - c$. Then define $\widetilde{x}' = \widetilde{x} - \chi_j e^j$. 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 ℓ_1-projection onto $\text{proj}_{bin}\, P$ that satisfies $\|u - \widetilde{x}\|_1 < \frac{1}{2}$.

Since $\text{AltProj}(\widetilde{x}) = \widetilde{x}$, Lemma 4.7 implies $\|\ell_1\text{-proj}(\widetilde{x}) - \widetilde{x}\|_1 \leq \frac{1}{2}$. Also, by Lemma 4.5 ℓ_1-proj$(\widetilde{x})$ has exactly one fractional coordinate, say, coordinate k (if ℓ_1-proj$(\widetilde{x})$ has *no* fractional coordinates, then $\widetilde{x} = \text{round}(\ell_1\text{-proj}(\widetilde{x})) = \ell_1\text{-proj}(\widetilde{x})$ and so $\widetilde{x} \in \text{proj}_{bin}\, P$, contradicting its definition); together these imply that $\widetilde{x} - \ell_1\text{-proj}(\widetilde{x}) = \alpha e^k$ for some $\alpha \in [-1/2, 1/2]$.

We claim that the coordinates k and j are different. To see this, notice that

$$a\widetilde{x} - c = a(\widetilde{x} - \ell_1\text{-proj}(\widetilde{x})) = \alpha a_k,$$

and since $|\alpha| \leq \frac{1}{2}$ this implies that $|a_k| \geq 2|a\widetilde{x} - c|$; by the definition of j, this is strictly greater than $|a_j|$, and thus $j \neq k$.

So define the point $u = \widetilde{x}' - \beta\alpha e^k$ for some β such that $au = c$; notice that such β exists and belongs to the interval $(0, 1)$, since at the bounds of this interval we get (using the definition of j)

$$a\widetilde{x}' = a(\widetilde{x} - \chi_j e^j) = a\widetilde{x} - a_j \succ c$$

and

$$a(\widetilde{x}' - \alpha e^k) = a(\ell_1\text{-proj}(\widetilde{x}) - \chi_j e^j) = c - a_j \prec c,$$

and since $a(\widetilde{x}' - \beta\alpha e^k)$ is continuous on β (so we can use the intermediate value theorem).

Notice that $u \in \text{proj}_{bin}\, P$: by construction $au = c$, $u_i \in [0, 1]$ for all $i \neq k$ (because $u_i = \widetilde{x}'_i$ and the right-hand side belongs to $\{0, 1\}$), and also $u_k \in [0, 1]$ (because we have the convex combination

$$u_k = (1 - \beta)\widetilde{x}'_k + \beta(\widetilde{x}' - \alpha) = (1 - \beta)\widetilde{x}_k + \beta(\widetilde{x}_k - \alpha)$$

and the terms $\widetilde{x}_k$ and $\widetilde{x}_k - \alpha = \ell_1\text{-proj}(\widetilde{x})_k$ belong to $[0, 1]$). Thus, u is indeed a candidate for ℓ_1-projection onto $\text{proj}_{bin}\, P$ and, hence,

$$\|\ell_1\text{-proj}(\widetilde{x}') - \widetilde{x}'\|_1 \leq \|u - \widetilde{x}'\|_1 = \beta|\alpha| < \frac{1}{2}.$$

Case 2: There is an index j with $a_j\chi_j \succ 0$ and $a_j\chi_j \succ a\widetilde{x} - c$. Given this hypothesis, there must be also an index k such that $a_k\chi_k \prec 0$, since from (5) we have that $\sum_i a_i\chi_i \preceq a\widetilde{x} - c$. To construct the vectors $\widetilde{x}'$ and u, consider the 2-dimensional system on variables α, β,

$$\begin{gathered} a\widetilde{x} - \alpha\, a_j\chi_j - \beta\, a_k\chi_k = c, \\ \alpha, \beta \in [0, 1]. \end{gathered}$$

This system is feasible, since setting $(\alpha,\beta)=(0,0)$ we obtain $a\widetilde{x}\succ c$ and setting $(\alpha,\beta)=(1,0)$ we obtain $a\widetilde{x}-a_j\chi_j\prec c$ (and as before we have continuity on α); in fact, this argument shows that there is a solution on the semiopen box $(0,1)\times[0,1]$. Moreover, because the terms $a_j\chi_j\succ 0$ and $a_k\chi_k\prec 0$ have opposite signs, the line $\{(\alpha,\beta)\in\mathbb{R}^2 : a\widetilde{x}-\alpha\, a_j\chi_j-\beta\, a_k\chi_k=c\}$ (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 $\{1\}\times[0,1]$ or the line $[0,1]\times\{1\}$. Thus, there is a solution $(\bar\alpha,\bar\beta)$ to the system where at least one of $\bar\alpha$ or $\bar\beta$ equals 1.

Now define $(\widetilde{\alpha},\widetilde{\beta})=\text{round}((\bar\alpha,\bar\beta))$, where again we round $\frac{1}{2}$ to 1. Then define $u=\widetilde{x}-\bar\alpha\chi_j e^j-\bar\beta\chi_k e^k$ and $\widetilde{x}'=\widetilde{x}-\widetilde{\alpha}\chi_j e^j-\widetilde{\beta}\chi_k e^k$. Clearly $\widetilde{x}'$ satisfies item 1 of the lemma: $\|\widetilde{x}'-\widetilde{x}\|_0\le 2$. Also, as before u is a candidate for ℓ_1-projection onto $\text{proj}_{bin}\,P$, which gives the ℓ_1 bound

$$\|\ell_1\text{-proj}(\widetilde{x}')-\widetilde{x}'\|_1\le\|u-\widetilde{x}'\|_1=\|(\bar\alpha,\bar\beta)-(\widetilde{\alpha},\widetilde{\beta})\|_1. \tag{6}$$

Recall at least one of $\bar\alpha$ or $\bar\beta$ 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 $\|\widetilde{x}'-x^*\|_0\le\|\widetilde{x}-x^*\|_0-1$ (because at least one of $\widetilde{\alpha}$ or $\widetilde{\beta}$ equals 1);
- if instead the other one equals $\frac{1}{2}$, then the right-hand side of (6) equals $\frac{1}{2}$ but both $\widetilde{\alpha}$ and $\widetilde{\beta}$ equal 1, thus $\|\widetilde{x}'-x^*\|_0\le\|\widetilde{x}-x^*\|_0-2$.

This concludes the proof. ∎

Now take the point $\widetilde{x}'$ closer to x^* than $\widetilde{x}$ constructed in the previous lemma and consider the effect of applying AltProj* to $\widetilde{x}'$. We would like to obtain that the final point AltProj$(\widetilde{x}')$ is still closer to x^* than where we began, i.e., we would like $\|\text{AltProj}^*(\widetilde{x}')-x^*\|_0\le\|\widetilde{x}-x^*\|_0-1$. If AltProj$^*(\widetilde{x}')=\widetilde{x}'$ 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 coordinatewise increase the vector $\widetilde{x}'$. Thus, if we compare the final vector AltProj$^*(\widetilde{x}')$ 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 $\|\widetilde{x}'-x^*\|_0\le\|\widetilde{x}-x^*\|_0-2$ of the lemma above when $\|\ell_1\text{-proj}(\widetilde{x}')-\widetilde{x}'\|_1=\frac{1}{2}$.

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

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

Now let k be the smallest such that AltProj$^k(\widetilde{x}')=$ AltProj$^*(\widetilde{x}')$, which exists from Corollary 4.8; so we want to show $\|w^k-x^*\|_0\le\|\widetilde{x}-x^*\|_0-1$. To simplify the notation, let $w^t\triangleq$ AltProj$^t(\widetilde{x}')$ for $t=0,1,\dots,k$. From Lemma 4.7 we have that $\|\ell_1\text{-proj}(w^t)-w^t\|_1\le\frac{1}{2}$ for all t's. Then the characterization of sequences of AltProj's of Lemma 4.10 give that there are indices $i_1,\dots,i_k$ such that

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

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

(7)
$$\|w^k - x^*\|_0 = \|w^0 - x^*\|_0 - [\#\ t\text{'s with } i_t \in \operatorname{supp}(x^*)] + [\#\ t\text{'s with } i_t \notin \operatorname{supp}(x^*)].$$

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

$$\#\ t\text{'s with } i_t \notin \operatorname{supp}(x^*) \leq \left\lceil \frac{k}{2} \right\rceil$$

(and the rest of the i_t's belong to $\operatorname{supp}(x^*)$). Employing these bounds on (7) we get

$$\|w^k - x^*\|_0 \leq \|w^0 - x^*\|_0 - \left\lfloor \frac{k}{2} \right\rfloor + \left\lceil \frac{k}{2} \right\rceil \leq \|w^0 - x^*\|_0 + 1. \qquad \square$$

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

Now going back to algorithm WFP-Compressed. Notice that since $\widetilde{\boldsymbol{z}}^\tau$ is obtained from AltProj$^*(.)$, it satisfies the fixed point condition AltProj$(\widetilde{\boldsymbol{z}}^\tau) = \widetilde{\boldsymbol{z}}^\tau$. Thus, as long as $\widetilde{\boldsymbol{z}}^\tau$ does not belong to $\operatorname{proj}_{bin} P$ we can apply the above lemma to obtain that, with probability at least $\frac{1}{n^2}$, the procedure RandWalkSAT$_2$ will flip coordinates of $\widetilde{\boldsymbol{z}}^\tau$ in a way that $\widetilde{\boldsymbol{z}}^{\tau+1} = \text{AltProj}^*(\text{RandWalkSAT}_2(\widetilde{\boldsymbol{z}}^\tau))$ is closer to x^* in ℓ_0 than the previous iterate $\widetilde{\boldsymbol{z}}^\tau$.

Corollary 4.13. *Let x^* be a coordinatewise maximal point in $\{0,1\}^n \cap \operatorname{proj}_{bin} P$. Then*

$$\Pr\left(\|\widetilde{\boldsymbol{z}}^{\tau+1} - x^*\|_0 \leq \|\widetilde{\boldsymbol{z}}^\tau - x^*\|_0 - 1 \;\middle|\; \widetilde{\boldsymbol{z}}^\tau \notin P\right) \geq \frac{1}{n^2}.$$

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 coordinatewise maximal point in $\{0,1\}^n \cap \operatorname{proj}_{bin} P$, and let $\boldsymbol{Z}_\tau = \|\widetilde{\boldsymbol{z}}^\tau - x^*\|_0$. Notice that $\boldsymbol{Z}_\tau = 0$ implies $\widetilde{\boldsymbol{z}}^\tau = x^*$ and hence $\widetilde{\boldsymbol{z}}^\tau \in P$, which implies that the algorithm stops. Corollary 4.13 gives that $\Pr(\boldsymbol{Z}_{\tau+1} \leq \boldsymbol{Z}_\tau - 1 \mid \widetilde{\boldsymbol{z}}^\tau \notin P) \geq \frac{1}{n^2}$. Therefore, if we start at iteration τ and for all the next $\boldsymbol{Z}_\tau$ iterations either the iterate $\widetilde{\boldsymbol{z}}^{\tau'}$ belongs to P or the algorithm reduces $\boldsymbol{Z}_{\tau'}$, it terminates by time $\tau + \boldsymbol{Z}_\tau$. Thus, with probability at least $(1/n^2)^{\boldsymbol{Z}_\tau} \geq (1/n^2)^n = p$ the algorithm terminates by time $\tau + \boldsymbol{Z}_\tau \leq \tau + n$.

Now let $\alpha = \lfloor T'/n \rfloor$ and call time steps $i \cdot n, \ldots, (i+1) \cdot n - 1$ the ith block of time. From the above paragraph, the probability that there is τ in the ith block of time such that $\widetilde{\boldsymbol{z}}^\tau \in P$ conditioned on $\widetilde{\boldsymbol{z}}^{i \cdot n - 1} \notin P$ is at least p. Using the chain rule of probability gives that the probability that there is no $\widetilde{\boldsymbol{z}}^\tau \in P$ within any of the

α 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 $\text{AltProj}^{n+1}(\widetilde{x}) = \text{AltProj}^*(\widetilde{x})$, it follows from inequality (4) that the original algorithm WFP terminates in at most $T' \cdot (n+1) = T$ iterations with probability at least $1-(1-p)^\alpha = 1-(1-p)^{\lfloor T/(n(n+1))\rfloor}$. This concludes the proof. ∎

4.2. Proof of Theorem 2.4. Fix a decomposable 1-row set P and let P_i denote its ith block, so $P = P_1 \times P_2 \times \cdots \times P_k$. Consider the execution of algorithm WFP over P, and let $\widetilde{\boldsymbol{x}}^t \in \{0,1\}^{n_1+\cdots+n_k}$ be the iterate produced by WFP at the end of iteration t. Let $\text{proj}_i : \mathbb{R}^{n_1+\cdots+n_k} \to \mathbb{R}^{n_i}$ denote the canonical projection to the coordinates corresponding to the ith block of P (so $\text{proj}_i\, \widetilde{\boldsymbol{x}}^t$ is the binary part of a tentative solution for the ith block).

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

Now for a block i, we define the (random) set of iterations where WFP modifies the ith block iterate $\text{proj}_i\, \widetilde{\boldsymbol{x}}^t$:

$$\mathbf{I}_i = \{t \geq 1 : \text{proj}_i\, \widetilde{\boldsymbol{x}}^t \neq \text{proj}_i\, \widetilde{\boldsymbol{x}}^{t-1}\}.$$

Consider a block i. Now we claim that the sequence $(\text{proj}_i\, \widetilde{\boldsymbol{x}}^t)_{t\in\mathbf{I}_i\cup\{0\}}$ 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 $\widetilde{\mathbf{x}}^t$, let $\widetilde{\mathbf{w}}^t \in \{0,1\}^{n_i}$ be the iterate at the end of iteration t when we apply WFP to the block P_i with starting point $\widetilde{w}^0 = \text{proj}_i\, \widetilde{x}^0$ (notice we used the letter $\mathbf{w}$ to replace the letter $\mathbf{x}$ 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* $(\text{proj}_i\, \widetilde{\boldsymbol{x}}^t)_{t\in\mathbf{I}_i\cup\{0\}}$ *and* $(\widetilde{\mathbf{w}}^t)_{t\geq 0}$ *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 followed by RandWalkSAT$_\ell$ (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 proj_i:

$$\text{proj}_i \circ \text{AltProj}_P = \text{AltProj}_{P_i} \circ \text{proj}_i\,.$$

Now we start comparing the sequences $(\text{proj}_i\, \widetilde{\boldsymbol{x}}^t)_{t\in\mathbf{I}_i\cup\{0\}}$ and $(\widetilde{\mathbf{w}}^t)_{t\geq 0}$ using a *coupling* argument. The idea is to show that if at some point both sequences have the same iterate, then the next items 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 jth smallest index in $\mathbf{I}_i$, denoted by $\mathbf{t}_j$, and assume by induction that $\text{proj}_i\, \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$ and $\widetilde{\mathbf{w}}^j$ have the same distribution; we couple them so as to have $\text{proj}_i\, \widetilde{\boldsymbol{x}}^{\mathbf{t}_j} = \widetilde{\mathbf{w}}^j$.

If $\text{proj}_i(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j})$ belongs to $\text{proj}_{bin} P_i$, then WFP_P will not change this part of the iterate anymore and RandWalkSAT_ℓ is not invoked in the ith block anymore (since the constraint in the ith block is satisfied it cannot be used in the minimal certificate). In this case, $\mathbf{t}_j$ is the last index in $\mathbf{I}_i$, i.e., $\widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$ is the last item of the sequence of iterates. Since $\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j} = \widetilde{\mathbf{w}}^j$, the same holds for $\widetilde{\mathbf{w}}^t$. Thus, there is no inductive step to be proved in this case.

Now suppose $\text{proj}_i(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j}) \notin \text{proj}_{bin} P_i$. We have two cases.

Case 1: $\text{AltProj}_{P_i}(\text{proj}_i(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j})) \neq \text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$. Then notice $\text{AltProj}_P(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j}) \neq \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$, since

$$\text{proj}_i \text{AltProj}_P(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j}) = \text{AltProj}_{P_i}(\text{proj}_i(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j})) \neq \text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}.$$

Thus, WFP_P changes the iterate in the iteration $\mathbf{t}_j+1$ and, hence, the next index in $\mathbf{I}_i$ is $\mathbf{t}_{j+1} = \mathbf{t}_j + 1$. Moreover, because of this change, the operator RandWalkSAT_ℓ is not invoked in this iteration of WFP_P and thus $\widetilde{\boldsymbol{x}}^{\mathbf{t}_{j+1}} = \text{AltProj}_P(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j})$, which implies that in the ith block $\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_{j+1}} = \text{AltProj}_{P_i}(\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j})$. The same observations hold for WFP_{P_i}, so

$$\widetilde{\mathbf{w}}^{t+1} = \text{AltProj}_{P_i} \widetilde{\mathbf{w}}^t = \text{AltProj}_{P_i}(\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}) = \text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_{j+1}},$$

proving the inductive step in this case.

Case 2: $\text{AltProj}_{P_i}(\text{proj}_i(\widetilde{\boldsymbol{x}}^{\mathbf{t}_j})) = \text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$. Because of this fixed-point property, the iterate $\text{proj}_i \widetilde{\boldsymbol{x}}^\tau$ remains the same for $\tau \in \{\mathbf{t}_j, \ldots, \mathbf{t}_{j+1} - 1\}$. Moreover, since $\mathbf{t}_{j+1} \in \mathbf{I}_i$ we have the iterate $\widetilde{\boldsymbol{x}}^{\mathbf{t}_{j+1}}$ is different from $\widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$; again because of the fixed-point property, it implies that at iteration $\mathbf{t}_{j+1}$ the algorithm WFP_P invokes RandWalkSAT_ℓ on block i. Thus, the iterate $\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_{j+1}}$ is obtained by applying RandWalkSAT_ℓ to $\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j}$ with the constraint of P_i as minimal projected certificate. For the same reason, algorithm WFP_{P_i} obtains $\widetilde{\mathbf{w}}^{t+1}$ by applying RandWalkSAT_ℓ to $\widetilde{\mathbf{w}}^t$ with the constraint of P_i as the minimal projected certificate. Since the initial points $\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_j} = \widetilde{\mathbf{w}}^t$ are the same, it follows that $\text{proj}_i \widetilde{\boldsymbol{x}}^{\mathbf{t}_{j+1}}$ and $\widetilde{\mathbf{w}}^{t+1}$ 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_{P_i} performs at most $(n_i(n_i+1) \cdot 2^{2n_i \log n_i} \cdot \lceil \ln(k/\delta) \rceil)$ iterations and, hence, by the equivalence from the above lemma this provides an upper bound on the length of the sequence $(\text{proj}_i \widetilde{\boldsymbol{x}}^t)_{t \in \mathbf{I}_i \cup \{0\}}$ or, equivalently, on the size of $\mathbf{I}_i$. Employing a union bound, with probability at least $1 - \delta$ we have that

$$\sum_{i=1}^{k} |\mathbf{I}_i| \leq \lceil \ln(k/\delta) \rceil \sum_{i=1}^{k} n_i(n_i+1) \cdot 2^{2n_i \log n_i}.$$

Since every iteration of algorithm WFP_P is accounted for in one of the sets $\mathbf{I}_i$, 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 in the tables, 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 Feasibility Pump. In hindsight, this can be justified by the following reasons:

- Picking a fixed ℓ 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 $|\widetilde{x}_j - \bar{x}_j|$. If $\mathbf{TT} \leq |F|$, 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 $(\widetilde{x}, \bar{y})$. We select the $|F|$ indices with largest fractionality $|\widetilde{x}_j - \bar{x}_j|$ and select uniformly at random $\min\{|S|, \mathbf{TT} - |F|\}$ indices from S, and flip the values in $\widetilde{x}$ 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 the case of a longer cycle, we use the very same restart strategy of Feasibility Pump.

5.2. Computational results. We tested the three algorithms (Feasibility Pump, 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 equivalents of two-stage stochastic programs and have the form

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

The variables x are the first-stage variables, and y^i are the second-stage variables for the ith 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 the 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 $\{0,1\}^{p+k\cdot q}$ 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 1 s 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 `wpertQ` the average percentage of WalkSAT-based perturbations. For completeness we also report in column `solver` the time CPLEX with default parameters took to solve the instances to optimality, though we believe the most relevant comparisons are between the different Feasibility Pump-type algorithms.

Table 1
Aggregated results by k on two-stage stochastic models.

	# found			itr.			time (s)			pgap			wpertQ	solver (s)
k	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	WFPB	
10	177	227	187	159	99	157	1.05	0.68	1.02	36%	20%	33%	16%	0.67
20	155	205	161	294	176	286	2.24	1.63	2.19	50%	39%	49%	30%	1.85
30	167	220	180	306	197	284	2.94	2.26	2.79	47%	36%	43%	34%	2.57
40	160	177	158	249	213	246	3.66	3.32	3.63	47%	43%	48%	18%	3.85
50	138	159	141	364	278	325	4.93	4.66	4.75	52%	48%	51%	23%	4.31

Then we aggregated the results based on seed. The corresponding results are reported in Table 2, where the last row provides average figures across seeds.

Table 2
Aggregated results by seed on two-stage stochastic models.

	# found			itr.			time (s)			pgap			wpertQ	solver (s)
seed	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	WFPB	
1	81	96	81	266	198	250	2.76	2.35	2.65	47%	39%	45%	22%	2.40
2	81	101	84	257	167	254	2.71	2.11	2.72	45%	36%	45%	26%	2.49
3	79	93	80	279	194	247	2.86	2.41	2.68	48%	40%	47%	25%	2.39
4	81	106	81	275	181	257	2.81	2.26	2.72	45%	35%	46%	23%	2.41
5	83	103	84	253	178	255	2.69	2.15	2.69	45%	35%	44%	25%	2.39
6	76	101	82	255	185	246	2.72	2.20	2.63	49%	37%	45%	27%	2.44
7	78	94	81	277	198	264	2.84	2.43	2.75	47%	39%	47%	27%	2.37
8	80	99	88	256	175	249	2.71	2.21	2.65	47%	37%	43%	25%	2.45
9	78	97	80	276	192	259	2.79	2.36	2.79	48%	37%	46%	26%	2.40
10	80	98	86	274	185	256	2.86	2.24	2.65	47%	38%	43%	24%	2.49
Avg.	80	99	83	267	185	254	2.78	2.27	2.69	47%	37%	45%	25%	2.42

On this testbed of models, both WFP and WFPB outperform Feasibility Pump; the first does so only marginally, while WFPB significantly improves over Feasibility Pump 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.

TABLE 3
Aggregated results by seed on MIPLIB 2010.

	# found			itr.			time (s)			pgap			wpertQ
seed	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	WFPB
1	279	280	272	43	43	55	8.24	8.32	9.88	48%	48%	50%	29%
2	279	279	275	44	44	56	8.40	8.33	10.03	50%	50%	52%	22%
3	277	285	270	43	41	55	8.32	8.02	9.80	48%	47%	50%	33%
4	280	282	271	42	41	53	8.07	7.89	9.38	48%	48%	51%	25%
5	276	277	271	42	41	54	8.26	8.21	9.82	51%	51%	52%	27%
6	277	278	270	43	42	55	8.29	8.13	9.76	50%	50%	52%	32%
7	278	281	274	43	41	53	8.17	8.04	9.65	50%	49%	51%	26%
8	273	277	269	43	43	52	8.16	8.07	9.15	49%	48%	51%	31%
9	282	282	275	42	41	52	8.13	7.95	9.48	49%	49%	52%	27%
10	278	282	275	42	40	53	8.33	8.02	9.79	50%	49%	52%	31%
Avg.	278	280	272	43	42	54	8.24	8.10	9.67	49%	49%	51%	28%

The improvement in this heterogeneous testbed is less dramatic than in the two-stage stochastic models. In this case, WFP performs consistently worst than Feasibility Pump, according to all measures. On the other hand, WFPB still consistently dominates Feasibility Pump, 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.

TABLE 4
Aggregated results by seed on hard models from MIPLIB 2010.

	# found			itr.			time (s)			pgap			wpertQ
seed	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	FP	WFPB	WFP	WFPB
1	119	120	112	191	194	281	17.76	18.09	23.78	47%	47%	52%	21%
2	119	119	115	202	200	286	18.34	18.09	24.21	48%	48%	51%	15%
3	118	125	110	194	179	280	17.93	16.92	23.39	48%	46%	51%	26%
4	120	122	111	185	177	268	17.41	16.78	22.77	46%	47%	51%	23%
5	116	117	111	188	183	276	17.09	16.90	23.04	49%	48%	51%	23%
6	117	118	110	190	186	276	17.85	17.23	23.22	50%	49%	53%	23%
7	118	121	114	191	180	263	17.50	17.03	22.83	48%	47%	50%	17%
8	113	117	109	196	193	261	17.75	17.42	21.68	50%	48%	52%	27%
9	122	122	115	189	179	255	17.23	16.60	22.00	48%	48%	51%	20%
10	118	122	115	189	174	265	17.36	16.21	22.61	47%	46%	50%	26%
Avg.	118	120	112	191	185	271	17.62	17.13	22.95	48%	47%	51%	22%

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

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 **TT**, 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 predetermined constant in the Feasibility Pump 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:

$$\begin{aligned} \max \quad & x_{T+2} \\ \text{s.t. } & 5x_1 + \cdots + 5x_{T+1} + 2x_{T+2} = 5T + 5, \\ & x_i \in \{0, 1\} \ \forall \ i \in [T + 2]. \end{aligned}$$

Clearly the LP optimal solution $\bar{x}^0$ is of the form $\bar{x}^0_{T+2} = 1$, $\bar{x}^0_i = \frac{3}{5}$ for some $i \in [T + 1]$ and $\bar{x}^0_j = 1$ for all $j \in [T + 1] \setminus \{i\}$. Rounding this we obtain $\widetilde{x}^0$, that is, the all 1's vector. It is also straightforward to verify that $\widetilde{x}^0$ is a stalling solution (i.e., $\text{AltProj}(\widetilde{x}^0) = \widetilde{x}^0$). So that algorithm randomly selects **TT** from the set $[t, T] \cap \mathbb{Z}$ and flips **TT** variables. Note that only one variable x_i (with $i \in [T + 1]$) has fractionality $|\frac{3}{5} - 1|$ and all other variables have fractionality 0. So using the convention for breaking ties, we flip x_i and **TT** $- 1$ other variables. Let S be the set of flipped variables and, since **TT** $\leq T < T + 1$, the variable x_{T+2} is not flipped. Thus, the point $\widetilde{x}$ obtained after slipping has $\widetilde{x}_{T+2} = 1$, $\widetilde{x}_j = 0$ for $j \in S$, and $\widetilde{x}_j = 1$ for $j \in [T + 1] \setminus S$.

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

1. if $S = \emptyset$, then $\widetilde{x}$ is again the stalling point $\widetilde{x}^0$;
2. if $S \neq \emptyset$, then $5\widetilde{x}_1 + \cdots + 5\widetilde{x}_{T+1} + 2\widetilde{x}_{T+2} < 5T + 5$ and after projecting to the LP relaxation we obtain a point of the form of $\bar{x}^0$ (i.e., exactly one of the coordinates $i \in [T + 1]$ equals $\frac{3}{5}$, all others equal 1). Rounding this solution again gives us the stalling point $\widetilde{x}^0$.

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

Appendix 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):*

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

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

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

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

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

since $(\bar{x}^2, \bar{y}^2) = \ell_1\text{-proj}(P, \tilde{x}^1)$, i.e., $\bar{x}^2$ is a closest point in ℓ_1-norm to $\tilde{x}^1$ in the projection of the LP relaxation in the x-space. Then

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

since $\tilde{x}^1$ and $\tilde{x}^2$ are both integer points and $\tilde{x}^2$ is obtained by rounding $\bar{x}^2$ (and a rounded point is a closest integer point in ℓ_1 norm). ∎

A *long cycle* in Feasibility Pump is a sequence

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

where

1. $(\bar{x}^i, \bar{y}^i)$, $i \in \{1, 2, \ldots, k\}$, are the vertices of the LP relaxation P, $\tilde{x}^i$, $i \in \{1, 2, \ldots, k\}$, are 0-1 vectors, $\tilde{x}^i = \text{round}(\bar{x}^i)$, and $(\bar{x}^{i+1}, \bar{y}^{i+1}) = \ell_1\text{-proj}(P, \tilde{x}^i)$;
2. $\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{k-1}$ are unique integer vectors;
3. $\bar{x}^1 = \bar{x}^k$, $\tilde{x}^1 = \tilde{x}^k$; 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 generality, 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 coordinatewise domination $\tilde{x}^{i+1} \geq \tilde{x}^i$, which contradicts this long cycle.

To show this domination, first notice that by Lemma B.1 the sequence of ℓ_1 gaps $\|\bar{x}^i - \tilde{x}^i\|_1$ is nondecreasing, and because of the cycle we have that the first and last terms of this sequence are the same; thus, all these gaps are the same. Hence, Lemma B.1 becomes equality: we have

$$\|\bar{x}^i - \tilde{x}^i\|_1 = \|\bar{x}^{i+1} - \tilde{x}^i\|_1 = \|\bar{x}^{i+1} - \tilde{x}^{i+1}\|_1$$

for all i. Letting J denote the set of indices j, where $\tilde{x}^i_j \neq \tilde{x}^{i+1}_j$, we can expand the last displayed equality to get

$$\begin{aligned} \sum_j |\tilde{x}^i_j - \bar{x}^{i+1}_j| &= \sum_{j \notin J} |\tilde{x}^i_j - \bar{x}^{i+1}_j| + \sum_{j \in J} |\tilde{x}^{i+1}_j - \bar{x}^{i+1}_j| \\ &\equiv \sum_{j \in J} |\tilde{x}^i_j - \bar{x}^{i+1}_j| = \sum_{j \in J} |\tilde{x}^{i+1}_j - \bar{x}^{i+1}_j|. \end{aligned} \tag{8}$$

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

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.

REFERENCES

[AB07] T. Achterberg and T. Berthold, *Improving the feasibility pump*, Discrete Optim., 4 (2007), pp. 77–86.

[Ach07] T. Achterberg, *Constraint Integer Programming*, Ph.D. thesis, Technische Universität Berlin, Berlin, 2007.

[BCLM09] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot, *A feasibility pump for mixed integer nonlinear programs*, Math. Program., 119 (2009), pp. 331–352.

[BEE+14] N. Boland, A. Eberhard, F. Engineer, M. Fischetti, M. Savelsbergh, and A. Tsoukalas, *Boosting the feasibility pump*, Math. Program. Comput., 6 (2014), pp. 255–279.

[BEET12] N. L. Boland, A. C. Eberhard, F. Engineer, and A. Tsoukalas, *A new approach to the feasibility pump in mixed integer programming*, SIAM J. Optim., 22 (2012), pp. 831–861.

[BFL07] L. Bertacco, M. Fischetti, and A. Lodi, *A feasibility pump heuristic for general mixed-integer problems*, Discrete Optim., 4 (2007), pp. 63–76.

[DFLL10] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi, *Experiments with a feasibility pump approach for nonconvex MINLPs*, in 9th International Symposium on Experimental Algorithms, SEA 2010, Springer, Berlin, 2010, pp. 350–360.

[DFLL12] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi, *A storm of feasibility pumps for nonconvex MINLP*, Math. Program., 136 (2012), pp. 375–402.

[DMW16] S. Dey, M. Molinaro, and Q. Wang, *Analysis of sparse cutting-planes for sparse MILPs with applications to stochastic MILPs*, Math. Oper. Res., to appear.

[FGL05] M. Fischetti, F. Glover, and A. Lodi, *The feasibility pump*, Math. Program., 104 (2005), pp. 91–104.

[FL11] M. Fischetti and A. Lodi, *Heuristics in mixed integer programming*, Wiley Encyclopedia of Operations Research and Management Science, Wiley, Hoboken, NJ, 2011.

[FS09] M. Fischetti and D. Salvagnin, *Feasibility pump* 2.0, Math. Program. Comput., 1 (2009), pp. 201–222.

[GMSS17] B. Geissler, A. Morsi, L. Schewe, and M. Schmidt, *Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps*, SIAM J. Optim., 27 (2017), pp. 1611–1636.

[ILO] IBM ILOG. *CPLEX oprtimizer*, http://www.ibm.com/software/integration/optimization/cplex/.

[KAA+11] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. D. Mittelmann, T. K. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter, MIPLIB 2010, Math. Program. Comput., 3 (2011), pp. 103–163.

[MJPL92] S. Minton, M. Johnston, A. Philips, and P. Laird, *Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems*, Artif. Intell., 58 (1992), pp. 161–205.

[MU05] M. Mitzenmacher and E.Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, New York, 2005.

[Pap91] C. Papadimitriou, *On selecting a satisfying truth assignment*, in FOCS, IEEE Computer Society, Los Alamitos, CA, pp. 163–169, 1991.

[Sch86] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, New York, 1986.

[Sch99] U. Schöning, *A probabilistic algorithm for k-sat and constraint satisfaction problems*, in Proceedings of the 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 1999, New York, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 410–414.

[SLR13] M. De Santis, S. Lucidi, and F. Rinaldi, *A new class of functions for measuring solution integrality in the feasibility pump approach*, SIAM J. Optim., 23 (2013), pp. 1575–1606.

[Tho00] H. Thorisson, *Coupling, Stationarity, and Regeneration*, Springer, Berlin, 2000.