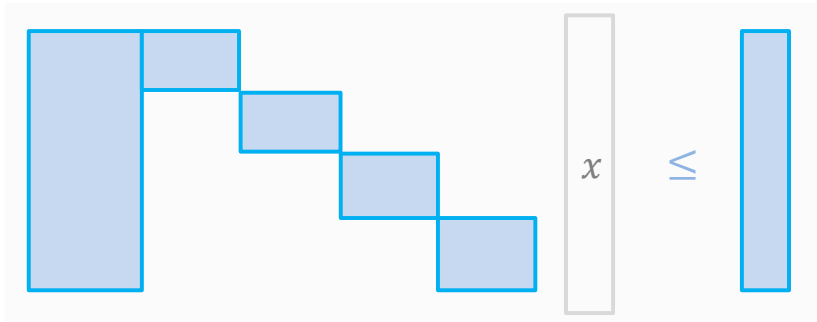
Improving the Randomization Step in Feasibility Pump using WalkSAT

Santanu S. Dey
Joint work with: Andres Iroume, Marco Molinaro, Domenico Salvagnin

Discrepancy & IP workshop, 2018

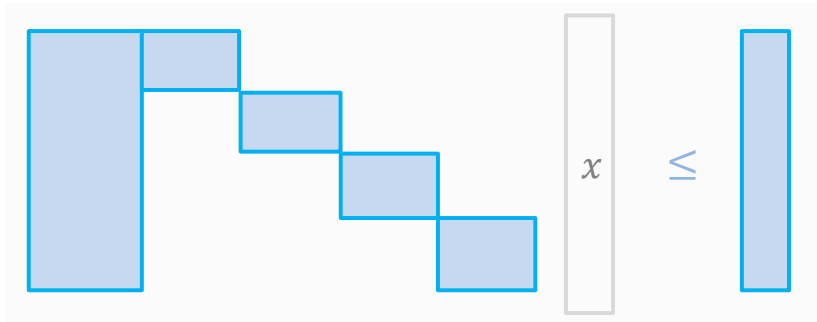
Sparsity in “real" Integer Programs (IPs)

- “Real" IPs are sparse: The average number (median) of non-zero entries in the constraint matrix of MIPLIB 2010 instances is 1.63% (0.17%).
- Many have "arrow shape" [Bergner, Caprara, Furini, Lübbecke, Malaguti, Traversi 11] or "almost decomposable structure" of the constraint matrix.
- Other example, two-stage Stochastic IPs:



Sparsity in “real" Integer Programs (IPs)

- “Real" IPs are sparse: The average number (median) of non-zero entries in the constraint matrix of MIPLIB 2010 instances is 1.63% (0.17%).
- Many have "arrow shape" [Bergner, Caprara, Furini, Lübbecke, Malaguti, Traversi 11] or "almost decomposable structure" of the constraint matrix.
- Other example, two-stage Stochastic IPs:



Goal: Exploit sparsity of IPs while designing primal heuristics, cutting-plane, branching rules...

1
Feasibility Pump

Introduction: Feasibility Pump (FP)

[Fischetti, Glover, Lodi 05]

Vanilla Feasibility Pump

- **Input:** Mixed-binary LP (with binary variables x and continuous variables y)
- Solve the linear programming relaxation, and let $(\bar{x}, \bar{y})$ be an optimal solution
- **While** $\bar{x}$ is not integral **do**:
  - Round: Round $\bar{x}$ to closest 0/1 values, call the obtained vector $\tilde{x}$.
  - Project: Let $(\bar{x}, \bar{y})$ be the point in the LP relaxation that minimizes $\sum_i |x_i - \tilde{x}_i|$ (we say, $\bar{x} = \ell_1\text{-proj}(\tilde{x})$).

Introduction: Feasibility Pump (FP)

[Fischetti, Glover, Lodi 05]

Vanilla Feasibility Pump

- **Input:** Mixed-binary LP (with binary variables x and continuous variables y)
- Solve the linear programming relaxation, and let $(\bar{x}, \bar{y})$ be an optimal solution
- **While** $\bar{x}$ is not integral **do**:
  - Round: Round $\bar{x}$ to closest 0/1 values, call the obtained vector $\tilde{x}$.
  - Project: Let $(\bar{x}, \bar{y})$ be the point in the LP relaxation that minimizes $\sum_i |x_i - \tilde{x}_i|$ (we say, $\bar{x} = \ell_1\text{-proj}(\tilde{x})$).

Problem: The above algorithm may cycle: Revisit the same $\tilde{x} \in \{0,1\}^n$ is different iterations (stalling).
Solution: Randomly perturb $\tilde{x}$.

Introduction: Feasibility Pump (FP)

[Fischetti, Glover, Lodi 05]

Vanilla Feasibility Pump

- **Input:** Mixed-binary LP (with binary variables x and continuous variables y)
- Solve the linear programming relaxation, and let $(\bar{x}, \bar{y})$ be an optimal solution
- **while** $\bar{x}$ is not integral do:
  - Round: Round $\bar{x}$ to closest 0/1 values, call the obtained vector $\tilde{x}$.
  - If stalling detected: Randomly perturb $\tilde{x}$ to a different 0/1 vector.
  - Project (ℓ_1-proj): Let $(\bar{x}, \bar{y})$ be the point in the LP relaxation that minimizes $\sum_i |x_i - \tilde{x}_i|$.

Feasibility Pump (FP)

- FP is very successful in practice (For example, the original FP finds feasible solutions for 96.3% of the instances in MIPLIB 2003 instances).
- Many improvements and generalizations: [Achterberg, Berthold 07], [Bertacco, Fischetti, Lodi 07], [Bonami, Cornuéjols, Lodi, Margot 09], [Fischetti, Salvagnin 09], [Boland, Eberhard, Engineer, Tsoukalas 12], [D'Ambrosio, Frangioni, Liberti, Lodi 12], [De Santis, Lucidi, Rinaldi 13], [Boland, Eberhard, Engineer, Fischetti, Savelsbergh, Tsoukalas 14], [Geißler, Morsi, Schewe, Schmidt 17], ...
- Some directions of research:
  - Take objective function into account
  - Mixed-integer programs with general integer variables.
  - Mixed-integer Non-linear programs (MINLP)
  - Alternative projection and rounding steps

Feasibility Pump (FP)

- FP is very successful in practice (For example, the original FP finds feasible solutions for 96.3% of the instances in MIPLIB 2003 instances).
- Many improvements and generalizations: [Achterberg, Berthold 07], [Bertacco, Fischetti, Lodi 07], [Bonami, Cornuéjols, Lodi, Margot 09], [Fischetti, Salvagnin 09], [Boland, Eberhard, Engineer, Tsoukalas 12], [D'Ambrosio, Frangioni, Liberti, Lodi 12], [De Santis, Lucidi, Rinaldi 13], [Boland, Eberhard, Engineer, Fischetti, Savelsbergh, Tsoukalas 14], [Geißler, Morsi, Schewe, Schmidt 17], ...
- Some directions of research:
  - Take objective function into account
  - Mixed-integer programs with general integer variables.
  - Mixed-integer Non-linear programs (MINLP)
  - Alternative projection and rounding steps

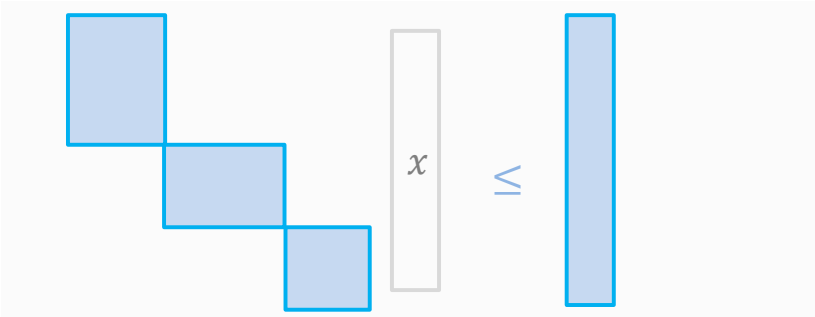
Randomization step plays significant role but has not been explicitly studied. We focus on changing the randomization step by "thinking about sparsity".

Sparse IPs $\approx$ Decomposable IPs

- As discussed earlier real integer programs are sparse.
- A common example of sparse integer programs is those that are almost decomposable.

Sparse IPs $\approx$ Decomposable IPs

- As discussed earlier real integer programs are sparse.
- A common example of sparse integer programs is those that are almost decomposable.
- As proxy, we keep in mind decomposable problems.



Agenda

- Propose a modification of WalkSAT for the mixed-binary case.
  - Show that this modified algorithm "works well" on-mixed-binary instances that are decomposable.

Agenda

- Propose a modification of WalkSAT for the mixed-binary case.
  - Show that this modified algorithm "works well" on-mixed-binary instances that are decomposable.
- Analyze randomization based on WalkSAT + Feasibility Pump.
  - Show that this version of FP "works well" on single-row decomposable instances.

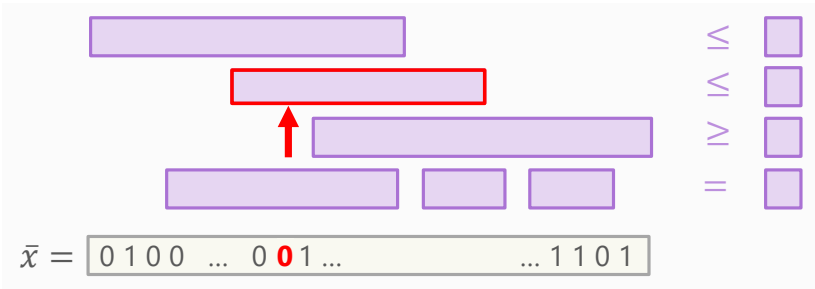
Agenda

- Propose a modification of WalkSAT for the mixed-binary case.
  - Show that this modified algorithm "works well" on-mixed-binary instances that are decomposable.
- Analyze randomization based on WalkSAT + Feasibility Pump.
  - Show that this version of FP "works well" on single-row decomposable instances.
- Implementation of FP with new randomization step that combines ideas from the previous randomization and new randomization.
  - The new method shows small but consistent improvement over FP.

2
WalkSAT

Introduction: WALKSAT

WalkSAT is effective primal heuristic used in SAT community [Schöning 99]



WalkSAT for pure binary IPs

- Start with a uniformly random point $\bar{x} \in \{0,1\}^n$. If feasible, done
- **While** $\bar{x}$ is infeasible **do**
  - Pick any violated constraint and randomly pick a variable $\bar{x}_i$ in its support
  - Flip value of $\bar{x}_i$

Performance of WalkSAT

- [Schöning 99] With probability $1 - \delta$, WALKSAT returns a feasible solution in $\sim \log(\frac{1}{\delta})2^n$ iterations.

Key Ideas:

Performance of WalkSAT

- [Schöning 99] With probability $1 - \delta$, WALKSAT returns a feasible solution in $\sim \log(\frac{1}{\delta})2^n$ iterations.

Key Ideas:

- Consider a fixed integer feasible solution x^*. Track the number of coordinates that are different from x^*.

Performance of WalkSAT

- [Schöning 99] With probability $1 - \delta$, WALKSAT returns a feasible solution in $\sim \log(\frac{1}{\delta})2^n$ iterations.

Key Ideas:

- Consider a fixed integer feasible solution x^*. Track the number of coordinates that are different from x^*.
- In each step, with probability at least

$$\underbrace{\frac{1}{s}}_{\text{\# non-zeros in violated constraint}}$$

we choose to flip coordinate where they differ.

Performance of WalkSAT

- [Schöning 99] With probability $1 - \delta$, WALKSAT returns a feasible solution in $\sim \log(\frac{1}{\delta})2^n$ iterations.

Key Ideas:

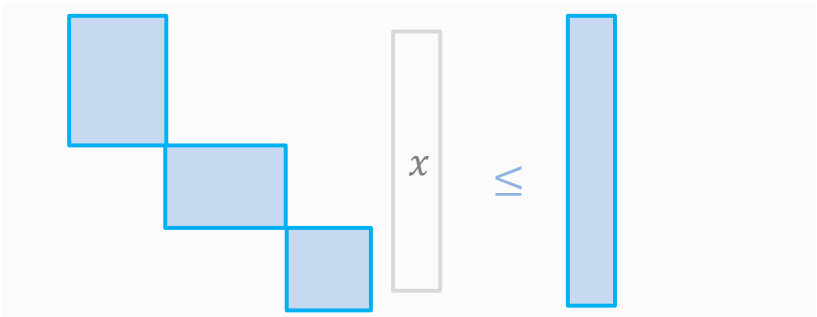
- Consider a fixed integer feasible solution x^*. Track the number of coordinates that are different from x^*.
- In each step, with probability at least

$$\underbrace{\frac{1}{s}}_{\text{\# non-zeros in violated constraint}}$$

we choose to flip coordinate where they differ.

- With probability atleast $\frac{1}{s}$ (a positive constant), reduce by 1 the number of coordinates they differ.

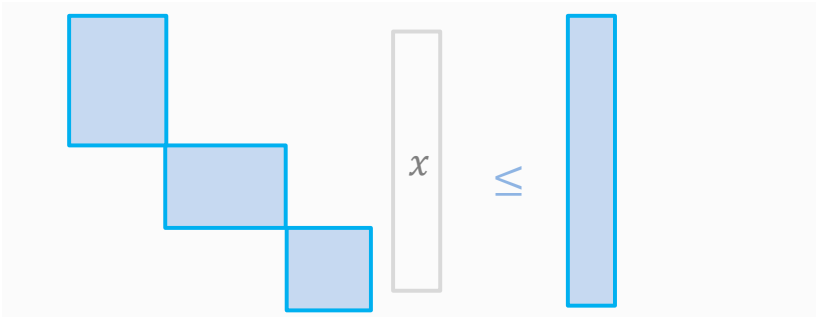
WalkSAT good for decomposable instances



Observation

- Each iteration depends only on one part. Overall execution can be split into independent executions over each part
- Put together bound from previous page over all parts

WalkSAT good for decomposable instances



Observation

- Each iteration depends only on one part. Overall execution can be split into independent executions over each part
- Put together bound from previous page over all parts

Consequences

- Find feasible solution in $\sim k2^{n/k}$ iterations.
- Compare this to total enumeration $\sim 2^n$ iterations.

3
Mixed-binary version of WalkSAT

Mixed-binary version of WalkSAT

WalkSAT(I) for Mixed-binary IPs

- **Input:** Mixed-binary LP-relaxation $\{(x, y) \mid Ax + By \leq b\}$ (with binary variables x and continuous variables y); parameter: I
- Start with a uniformly random point $\bar{x} \in \{0, 1\}^n$. If $\exists \bar{y}$ such that $(\bar{x}, \bar{y})$ is feasible, done
- **While** $\bar{x} \notin \text{Proj}_x(P)$ **do**
  - Generate minimal (wrt support of λ) projected certificate of infeasibility:

$$\lambda^\top Ax \leq \lambda^\top b$$

    1. a valid inequality for $\text{Proj}_x(P)$ i.e.,$\lambda \geq 0$, $\lambda^\top B = 0$.
    2. violating $\bar{x}$: $(\lambda^\top A)\bar{x} > \lambda^\top b$

    (Can be obtained by solving a LP)
  - Randomly pick I variables (with replacement) in the support of minimal projected certificate.
  - Flip value of these variables

Mixed-binary WalkSAT

Key Observation: If a set is decomposable, then minimal certificate has a support contained in exactly one disjoint set of variables.

Mixed-binary WalkSAT

Key Observation: If a set is decomposable, then minimal certificate has a support contained in exactly one disjoint set of variables.

Theorem

Consider a feasible decomposable mixed-binary set

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

Let s_i be such that each constraint in P_i has at most s_i binary variables, and define $\gamma_i := \min\{s_i \cdot (d_i + 1), n_i\}$. Then with probability at least $1 - \delta$, Mixed-binary WalkSAT(1) returns a feasible solution within

$$\ln(k/\delta) \sum_i n_i 2^{n_i \log \gamma_i}$$

iterations.

4
Feasibility Pump + WalkSAT

FP + WalkSAT (FPW)

Vanilla Feasibility Pump

- **Input:** Mixed-binary LP (with binary variables x and continuous variables y)
- Solve the linear programming relaxation, and let $(\bar{x}, \bar{y})$ be an optimal solution
- **while** $\bar{x}$ is not integral do:
  - Round: Round $\bar{x}$ to closest 0/1 values, call the obtained vector $\tilde{x}$.
  - If Stalling detected: "Randomly" perturb $\tilde{x}$ to a different 0/1 vector.
  - Project: Let $(\bar{x}, \bar{y})$ be the point in the LP relaxation that minimizes $\sum_i |x_i - \tilde{x}_i|$.

FP + WalkSAT (FPW)

Feasibility Pump + WalkSAT

- **Input:** Mixed-binary LP (with binary variables x and continuous variables y)
- Solve the linear programming relaxation, and let $(\bar{x}, \bar{y})$ be an optimal solution
- **while** $\bar{x}$ is not integral do:
  - Round: Round $\bar{x}$ to closest 0/1 values, call the obtained vector $\tilde{x}$.
  - If Stalling detected: "Randomly" perturb $\tilde{x}$ to a different 0/1 vector. $<---$ Use mixed-binary WalkSAT(I) for random update
  - Project: Let $(\bar{x}, \bar{y})$ be the point in the LP relaxation that minimizes $\sum_i |x_i - \tilde{x}_i|$.

Analysis of Feasibility Pump + WalkSAT (FPW)

1. We are not able to analyze this algorithm WFP for a general mixed-binary IP
   - Issue: From previous proof, with probability $1/s_j$ randomization makes progress, but projection+rounding in next iteration could ruin everything

Analysis of Feasibility Pump + WalkSAT (FPW)

1. We are not able to analyze this algorithm WFP for a general mixed-binary IP
   - Issue: From previous proof, with probability $1/s_j$ randomization makes progress, but projection+rounding in next iteration could ruin everything
2. Can analyze running-time for decomposable 1-row instances, i.e. instances of the following kind:

$$\left.\begin{array}{l} a^i x^i + b^i y^i = c_i \\ x^i \in \{0,1\}^{n_i}, y_i \in \mathbb{R}_+^{d_i}. \end{array}\right\} \ \forall i \in [k]$$

Main result

Theorem

Consider a feasible decomposable 1-row instances set as shown in the previous slide. Then with probability at least $1 - \delta$*, Feasibility Pump + WalkSAT(2) returns a feasible solution within*

$$T = \lceil \ln(k/\delta) \rceil \sum_{i \in [k]} n_i(n_i + 1) \cdot 2^{2n_i \log n_i} \leq \lceil \ln(k/\delta) \rceil \, k(\bar{n} + 1)^2 \cdot 2^{2\bar{n} \log \bar{n}}$$

iterations, where $\bar{n} = \max_i n_i$*.*

Main result

Theorem

Consider a feasible decomposable 1-row instances set as shown in the previous slide. Then with probability at least $1 - \delta$*, Feasibility Pump + WalkSAT(2) returns a feasible solution within*

$$T = \lceil \ln(k/\delta) \rceil \sum_{i \in [k]} n_i(n_i + 1) \cdot 2^{2n_i \log n_i} \leq \lceil \ln(k/\delta) \rceil \, k(\bar{n} + 1)^2 \cdot 2^{2\bar{n} \log \bar{n}}$$

iterations, where $\bar{n} = \max_i n_i$*.*

Note: Naive Feasibility Pump with original randomization (and no re-start) may fail to converge for these instances.

Proof sketch - I

- (Like before) We can split the execution into independent executions over each constraint.

Proof sketch - I

- (Like before) We can split the execution into independent executions over each constraint.
- Notation: For $\tilde{x} \in \{0,1\}^n$: AltProj$(\tilde{x}) :=$ round $(\ell_1\text{-proj}(\tilde{x}))$.

Proof sketch - I

- (Like before) We can split the execution into independent executions over each constraint.
- Notation: For $\tilde{x} \in \{0,1\}^n$: AltProj$(\tilde{x}) :=$ round $(\ell_1\text{-proj}(\tilde{x}))$.

Proposition (Length of cycle)

All cycles are due to short cycles, i.e. randomization is invoked only when AltProj$(\tilde{x}) = \tilde{x}$.

Proof sketch - I

- (Like before) We can split the execution into independent executions over each constraint.
- Notation: For $\tilde{x} \in \{0,1\}^n$: AltProj$(\tilde{x}) :=$ round $(\ell_1\text{-proj}(\tilde{x}))$.

Proposition (Length of cycle)

All cycles are due to short cycles, i.e. randomization is invoked only when AltProj$(\tilde{x}) = \tilde{x}$.

- Notation (Stabilization): AltProj$^*(\tilde{x}) = \bar{x}$, where AltProj$^k(\tilde{x}) =$ AltProj$^{k+1}(\tilde{x}) = \bar{x}$ for some $k \in \mathbb{Z}_{++}$.

Proof sketch - I

- (Like before) We can split the execution into independent executions over each constraint.
- Notation: For $\tilde{x} \in \{0,1\}^n$: $\text{AltProj}(\tilde{x}) := \text{round}\,(\ell_1\text{-proj}(\tilde{x}))$.

Proposition (Length of cycle)

All cycles are due to short cycles, i.e. randomization is invoked only when $\text{AltProj}(\tilde{x}) = \tilde{x}$.

- Notation (Stabilization): $\text{AltProj}^*(\tilde{x}) = \bar{x}$, where $\text{AltProj}^k(\tilde{x}) = \text{AltProj}^{k+1}(\tilde{x}) = \bar{x}$ for some $k \in \mathbb{Z}_{++}$.
-

$$\#\text{ of iterations FPW} \leq \underbrace{[\#\text{ iterations of AltProj}^*]}_{\text{Number of stallings}} \times \underbrace{\max_{\tilde{x}\in\{0,1\}^n} \min\{k : \text{altProj}^k(\tilde{x}) = \text{altProj}^*(\tilde{x})\}}_{\text{Worst-case stabilization time}}.$$

Proof sketch - I

- (Like before) We can split the execution into independent executions over each constraint.
- Notation: For $\tilde{x} \in \{0,1\}^n$: $\text{AltProj}(\tilde{x}) := \text{round}\,(\ell_1\text{-proj}(\tilde{x}))$.

Proposition (Length of cycle)

All cycles are due to short cycles, i.e. randomization is invoked only when $\text{AltProj}(\tilde{x}) = \tilde{x}$.

- Notation (Stabilization): $\text{AltProj}^*(\tilde{x}) = \bar{x}$, where $\text{AltProj}^k(\tilde{x}) = \text{AltProj}^{k+1}(\tilde{x}) = \bar{x}$ for some $k \in \mathbb{Z}_{++}$.
-

$$\#\text{ of iterations FPW} \leq \underbrace{[\#\text{ iterations of AltProj}^*]}_{\text{Number of stallings}} \times \underbrace{\max_{\tilde{x}\in\{0,1\}^n} \min\{k : \text{altProj}^k(\tilde{x}) = \text{altProj}^*(\tilde{x})\}}_{\text{Worst-case stabilization time}}.$$

Proposition (Worst-case stabilization time)

For any $\tilde{x} \in \{0,1\}^n$, $\text{AltProj}^{n+1}(\tilde{x}) = \text{AltProj}(\tilde{x})$.

Proof sketch - II

$$\begin{aligned} [x^2 := \text{AltProj}^*(\tilde{x}^1)] &\longrightarrow [\tilde{x}^2 := \text{WALKSAT}(x^2)] \\ \longrightarrow [x^3 := \text{AltProj}^*(\tilde{x}^2)] &\longrightarrow [\tilde{x}^3 := \text{WALKSAT}(x^3)] \\ \longrightarrow [x^4 := \text{AltProj}^*(\tilde{x}^3)] &\longrightarrow [\tilde{x}^4 := \text{WALKSAT}(x^4)] \ldots \end{aligned}$$

Like last time, we target a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$.

Proof sketch - II

$$\begin{aligned} [x^2 := \text{AltProj}^*(\tilde{x}^1)] &\longrightarrow [\tilde{x}^2 := \text{WALKSAT}(x^2)] \\ \longrightarrow [x^3 := \text{AltProj}^*(\tilde{x}^2)] &\longrightarrow [\tilde{x}^3 := \text{WALKSAT}(x^3)] \\ \longrightarrow [x^4 := \text{AltProj}^*(\tilde{x}^3)] &\longrightarrow [\tilde{x}^4 := \text{WALKSAT}(x^4)] \ldots \end{aligned}$$

Like last time, we target a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$. Key Question:

- What if we get closer to x^* in the WalkSAT step, but then **go far away** in the AltProj* step.

Proof sketch - II

$$\begin{aligned} [x^2 := \text{AltProj}^*(\tilde{x}^1)] &\longrightarrow [\tilde{x}^2 := \text{WALKSAT}(x^2)] \\ \longrightarrow [x^3 := \text{AltProj}^*(\tilde{x}^2)] &\longrightarrow [\tilde{x}^3 := \text{WALKSAT}(x^3)] \\ \longrightarrow [x^4 := \text{AltProj}^*(\tilde{x}^3)] &\longrightarrow [\tilde{x}^4 := \text{WALKSAT}(x^4)] \ldots \end{aligned}$$

Like last time, we target a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$. Key Question:

- What if we get closer to x^* in the WalkSAT step, but then **go far away** in the AltProj* step.

Lemma

Consider a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$*, and a point* $\tilde{x} \in \{0,1\}^n$ *not in* $\text{Proj}_x(P) \cap \{0,1\}^n$*. Suppose* $\text{altProj}(\tilde{x}) = \tilde{x}$*.*

Proof sketch - II

$$[x^2 := \text{AltProj}^*(\tilde{x}^1)] \longrightarrow [\tilde{x}^2 := \text{WALKSAT}(x^2)]$$
$$\longrightarrow [x^3 := \text{AltProj}^*(\tilde{x}^2)] \longrightarrow [\tilde{x}^3 := \text{WALKSAT}(x^3)]$$
$$\longrightarrow [x^4 := \text{AltProj}^*(\tilde{x}^3)] \longrightarrow [\tilde{x}^4 := \text{WALKSAT}(x^4)] \ldots$$

Like last time, we target a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$. Key Question:

- What if we get closer to x^* in the WalkSAT step, but then **go far away** in the AltProj* step.

Lemma

Consider a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$, and a point $\tilde{x} \in \{0,1\}^n$ not in $\text{Proj}_x(P) \cap \{0,1\}^n$. Suppose $\text{altProj}(\tilde{x}) = \tilde{x}$. *Then there is a point $\tilde{x}' \in \{0,1\}^n$ satisfying the following:*

1. *(close to $\tilde{x}$)* $\|\tilde{x}' - \tilde{x}\|_0 \leq 2$
2. *(closer to x^*)* $\|\tilde{x}' - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1$

Proof sketch - II

$$[x^2 := \text{AltProj}^*(\tilde{x}^1)] \longrightarrow [\tilde{x}^2 := \text{WALKSAT}(x^2)]$$
$$\longrightarrow [x^3 := \text{AltProj}^*(\tilde{x}^2)] \longrightarrow [\tilde{x}^3 := \text{WALKSAT}(x^3)]$$
$$\longrightarrow [x^4 := \text{AltProj}^*(\tilde{x}^3)] \longrightarrow [\tilde{x}^4 := \text{WALKSAT}(x^4)] \ldots$$

Like last time, we target a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$. Key Question:

- What if we get closer to x^* in the WalkSAT step, but then **go far away** in the AltProj* step.

Lemma

Consider a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$, and a point $\tilde{x} \in \{0,1\}^n$ not in $\text{Proj}_x(P) \cap \{0,1\}^n$. Suppose $\text{altProj}(\tilde{x}) = \tilde{x}$. *Then there is a point $\tilde{x}' \in \{0,1\}^n$ satisfying the following:*

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

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

Proof sketch - II

$$\begin{aligned} & [x^2 := \text{AltProj}^*(\tilde{x}^1)] \quad \longrightarrow \quad [\tilde{x}^2 := \text{WALKSAT}(x^2)] \\ \longrightarrow\ & [x^3 := \text{AltProj}^*(\tilde{x}^2)] \quad \longrightarrow \quad [\tilde{x}^3 := \text{WALKSAT}(x^3)] \\ \longrightarrow\ & [x^4 := \text{AltProj}^*(\tilde{x}^3)] \quad \longrightarrow \quad [\tilde{x}^4 := \text{WALKSAT}(x^4)] \ldots \end{aligned}$$

Like last time, we target a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$. Key Question:

- What if we get closer to x^* in the WalkSAT step, but then **go far away** in the AltProj* step.

Lemma

Consider a point $x^* \in \text{Proj}_x(P) \cap \{0,1\}^n$, and a point $\tilde{x} \in \{0,1\}^n$ not in $\text{Proj}_x(P) \cap \{0,1\}^n$. Suppose $\text{altProj}(\tilde{x}) = \tilde{x}$. *Then there is a point $\tilde{x}' \in \{0,1\}^n$ satisfying the following:*

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

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

Corollary

Let x^* be a coordinate-wise maximal solution in $\{0,1\}^n \cap \text{Proj}_x(P)$. Consider any point $\tilde{x} \in \{0,1\}^n \setminus \text{Proj}_x(P)$ satisfying $\text{altProj}^*(\tilde{x}) = \tilde{x}$, *and let $\tilde{x}' \in \{0,1\}^n$ be a point constructed in Lemma above with respect to x^* and $\tilde{x}$. Then* $\|\text{altProj}^*(\tilde{x}') - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1$.

5
Computations

Proposed randomization

- All features (such as constraint propagation) which are part of the Feasibility Pump 2.0 ([Fischetti, Salvagnin 09]) code have been left unchanged.
- The only change is in the randomization step.

Proposed randomization

- All features (such as constraint propagation) which are part of the Feasibility Pump 2.0 ([Fischetti, Salvagnin 09]) code have been left unchanged.
- The only change is in the randomization step.

Old randomization

- Define fractionality of i^{th} variable: $|\bar{x}_i - \tilde{x}_i|$. Let F be the number of variables with positive fractionality.
- Randomly generate an integer TT (uniformly from $\{10, \ldots, 30\}$).
- Flip the $\min\{F, TT\}$ variables with highest fractionality.

Proposed randomization

- All features (such as constraint propagation) which are part of the Feasibility Pump 2.0 ([Fischetti, Salvagnin 09]) code have been left unchanged.
- The only change is in the randomization step.

Old randomization

- Define fractionality of i^{th} variable: $|\bar{x}_i - \tilde{x}_i|$. Let F be the number of variables with positive fractionality.
- Randomly generate an integer TT (uniformly from $\{10, \ldots, 30\}$).
- Flip the $\min\{F, TT\}$ variables with highest fractionality.

New randomization

- Flip the $\min\{F, TT\}$ variables with highest fractionality.
- If $F < TT$, then:
  - let S be the union of the supports of the constraints that are not satisfied by the current point $(\tilde{x}, \bar{y})$.
  - Select uniformly at random $\min\{|S|, TT - |F|\}$ indices from S, and flip the values in $\tilde{x}$ for all the selected indices.

Computational experiments

Two classes of problems:

1. Two-stage stochastic models (randomly generated)

$$Ax + D^i y^i \leq b^i,\ i \in \{1, \ldots, k\}$$
$$x \in \{0,1\}^p$$
$$y^i \in \{0,1\}^q\ \ i \in \{1, \ldots, k\}$$

2. MIPLIB 2010

Two algorithms:

1. FP: Feasibility pump 2.0
2. FPWM: Feasibility pump 2.0 + the modified randomization above

Results for stochastic instances

	# found		itr.		time (s)		% gap		% modified
seed	FP	FPWM	FP	FPWM	FP	FPWM	FP	FPWM	FPWM
1	81	96	266	198	2.76	2.35	47%	39%	22%
2	81	101	257	167	2.71	2.11	45%	36%	26%
3	79	93	279	194	2.86	2.41	48%	40%	25%
4	81	106	275	181	2.81	2.26	45%	35%	23%
5	83	103	253	178	2.69	2.15	45%	35%	25%
6	76	101	255	85	2.72	2.20	49%	37%	27%
7	78	94	277	198	2.84	2.43	47%	39%	27%
8	80	99	256	175	2.71	2.21	47%	37%	25%
9	78	97	276	192	2.79	2.36	48%	37%	26%
10	80	98	274	185	2.86	2.24	47%	38%	24%
Avg.	80	99	267	185	2.78	2.27	47%	37%	25%

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

- 150 instances
- $k \in \{10, 20, 30, 40, 50\}$
- $p = q \in \{10, 20\}$

Results for MIPLIB 2010 instances

	# found		itr.		time (s)		% gap		% modified
seed	FP	FPWM	FP	FPWM	FP	FPWM	FP	FPWM	FPWM
1	279	280	43	43	8.24	8.32	48%	48%	29%
2	279	279	44	44	8.40	8.33	50%	50%	22%
3	277	285	43	41	8.32	8.02	48%	47%	33%
4	280	282	42	41	8.07	7.89	48%	48%	25%
5	276	277	42	41	8.26	8.21	51%	51%	27%
6	277	278	43	42	8.29	8.13	50%	50%	32%
7	278	281	43	41	8.17	8.04	50%	49%	26%
8	273	277	43	43	8.16	8.07	49%	48%	31%
9	282	282	42	41	8.13	7.95	49%	49%	27%
10	278	282	42	40	8.33	8.02	50%	49%	31%
Avg.	278	280	43	42	8.24	8.10	49%	49%	28%

Table: Aggregated results by seed on MIPLIB2010.

Conclusions

- First ever analysis of running time of Feasibility Pump (even if it is for a special class of instances)
- Suggested changes are trivial to implement and appears to dominate feasibility pump almost consistently.

Conclusions

- First ever analysis of running time of Feasibility Pump (even if it is for a special class of instances)
- Suggested changes are trivial to implement and appears to dominate feasibility pump almost consistently.
- "Designing for sparse instances" helps!