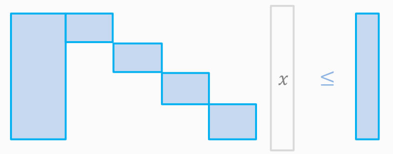
Using Sparsity to Design Primal Heuristics for MILPs: Two Stories

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

MIP Workshop, 2017

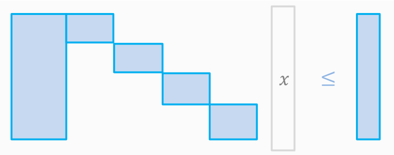
Sparsity in “real" Integer Programs (IPs)

- “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?

1

Story 1: New randomization step for Feasibility Pump

Joint work with: Andres Iroume[1], Marco Molinaro[2], and Domenico Salvagnin[3]

[1] Revenue Analytics, USA

[2] PUC-Rio, Brazil

[3] IBM Italy and DEI, University of Padova, Italy

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

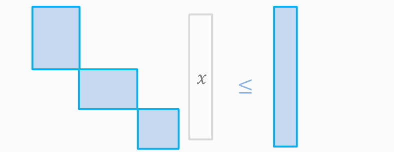
Randomization step plays significant role but has not been explicitly studied. We focus on changing the randomization step by "thinking about sparsity".

Sparse IPs ≈ 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

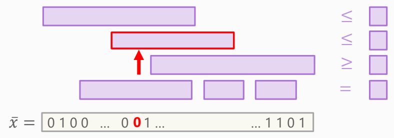
- Propose a modification of WalkSAT for the mixed-binary case.
  - Show that this modified algorithm "works well" on-mixed-binary instances that are decomposable.
- Analyze randomization based on WalkSAT + Feasibility Pump.
  - Show that this version of FP "works well" on single-row decomposable instances.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

1.1

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] WalkSAT returns a feasible solution in $\sim 2^n$ iterations, in expectation.

Key Ideas:

Performance of WalkSAT

- [Schöning 99] WalkSAT returns a feasible solution in $\sim 2^n$ iterations, in expectation.

Key Ideas:

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

Performance of WalkSAT

- [Schöning 99] WalkSAT returns a feasible solution in $\sim 2^n$ iterations, in expectation.

Key Ideas:

- Consider a fixed integer feasible solution x^*. Track the number of coordinates that 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] WalkSAT returns a feasible solution in $\sim 2^n$ iterations, in expectation.

Key Ideas:

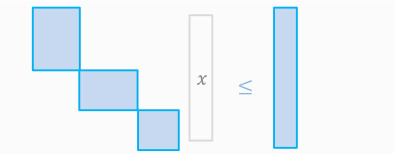
- Consider a fixed integer feasible solution x^*. Track the number of coordinates that different from x^*.
- In each step, with probability at least

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

we choose to flip coordinate where they differ

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

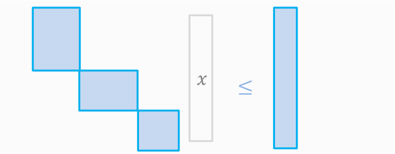
WalkSAT good for decomposable instances



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, in expectation.
- Compare this to total enumeration $\sim 2^n$ iterations.

1.2
Mixed-binary version of WalkSAT

Mixed-binary version of WalkSAT

WalkSAT(l) for Mixed-binary IPs

- **Input:** Mixed-binary LP-relaxation $\{(x, y) \,|\, Ax + By \leq b\}$ (with binary variables x and continuous variables y); parameter: l
- 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** $\bar{x} \notin \text{Proj}_x(P)$ is infeasible **do**
  - Generate minimal 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 $\bar{x}$: $(\lambda^\top A)\bar{x} > \lambda^\top b$
    3. minimal with respect to support of λ.

    (Can be obtained by solving a LP)
  - Randomly pick l 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

$$P^I = P^I_1 \times \ldots \times P^I_k, \text{ where for all } i \in [k] \text{ we have}$$
$$P^I_i = P_i \cap (\{0,1\}^{n_i} \times \mathbb{R}^{d_i}),$$
$$P_i = \{(x^i, y^i) \in [0,1]^{n_i} \times \mathbb{R}^{d_i} : A^i x^i + B^i y^i \leq c^i\}. \tag{1}$$

Let 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.

1.3
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:

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

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 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 $\tilde{x} \in \{0, 1\}^n$: $\text{AltProj}(\tilde{x}) := \text{round}\left(\ell_1\text{-proj}(\tilde{x})\right)$.

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 $\text{AltProj}(\tilde{x}) = \tilde{x}$.

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 $k \in \mathbb{Z}_{++}$.

Proof sketch - I

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

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

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

Like last time, we target a point $x^* \in \mathrm{Proj}_x(P) \cap \{0,1\}^n$. 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 $x^* \in \mathrm{Proj}_x(P) \cap \{0,1\}^n$*, and a point* $\tilde{x} \in \{0,1\}^n$ *not in* $\mathrm{Proj}_x(P) \cap \{0,1\}^n$*. Suppose* $\mathrm{altProj}(\tilde{x}) = \tilde{x}$*.*

Proof sketch - II

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

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

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

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 $\text{altProj}(\tilde{x}) = \tilde{x}$. 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

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

Like last time, we target a point $x^* \in \mathrm{Proj}_x(P) \cap \{0,1\}^n$. 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 $x^* \in \mathrm{Proj}_x(P) \cap \{0,1\}^n$, and a point $\tilde{x} \in \{0,1\}^n$ not in $\mathrm{Proj}_x(P) \cap \{0,1\}^n$. Suppose $\mathrm{altProj}(\tilde{x}) = \tilde{x}$. 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 $\{0,1\}^n \cap \mathrm{Proj}_x(P)$. Consider any point $\tilde{x} \in \{0,1\}^n \setminus \mathrm{Proj}_x(P)$ satisfying $\mathrm{altProj}^*(\tilde{x}) = \tilde{x}$, and let $\tilde{x}' \in \{0,1\}^n$ be a point constructed in Lemma above with respect to x^* and $\tilde{x}$. Then $\|\mathrm{altProj}^*(\tilde{x}') - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1$.

1.4 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 $\{10, \dots, 30\}$).
- 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)

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

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!

2

Story 2: Approximate algorithm for sparse packing problems

Joint work with: Qianyi Wang[4] and Marco Molinaro[5]

---

[4] Uber Inc.

[5] PUC-Rio, Brazil

Packing instances

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}_+^n, \end{aligned}$$

where A and b is non-negative.

Packing instances

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_+, \end{aligned}$$

where A and b is non-negative.

High-level sketch of algorithm

The algorithm runs in two phases.

Packing instances

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_+, \end{aligned}$$

where A and b is non-negative.

High-level sketch of algorithm

The algorithm runs in two phases.

1. In the first phase, the sparse packing problem is partitioned into smaller parts and then these small integer programs are solved.

Packing instances

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_+, \end{aligned}$$

where A and b is non-negative.

High-level sketch of algorithm

The algorithm runs in two phases.

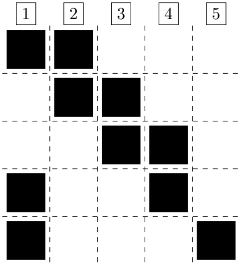
1. In the first phase, the sparse packing problem is partitioned into smaller parts and then these small integer programs are solved.
2. In the second phase, the optimal solutions of the smaller problems are patched together into a feasible solution for the original problem by exploiting the sparsity structure of the constraint matrix.

2.1
Key ideas

Key idea - I

Suppose the matrix A looks like this:

- 5 blocks of variables.
- Unshaded boxes correspond to zeros in A.



$$A^1x^1 + A^2x^2 + A^3x^3 + Ax^4 + A^5x^5 \leq b, \;\; x^i \in \mathbb{Z}_+^{n_i}$$

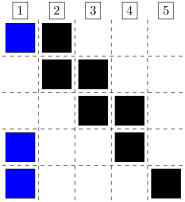
Key idea - I

- Suppose we fix variables in blocks $\{2, 3, 4, 5\}$ to zero and find a non-zero feasible solution in variables of block 1.

$$0$$

$$\mathbf{A^1x^1} + A^2x^2 + A^3x^3 + Ax^4 + A^5x^5 \leq \mathbf{b},\ x^i \in \mathbb{Z}_+^{n_i}$$

- Let this solutions be: $(\bar{x}^1, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$

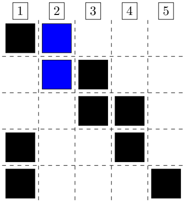


Key idea - I

- Similarly we can fix variables in blocks $\{1, 3, 4, 5\}$ to zero and find a non-zero feasible solution in variables of block 2.

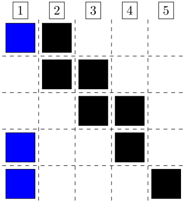
$$\cancelto{0}{A^1x^1} + \mathbf{A^2x^2} + A^3x^3 + \cancelto{0}{Ax^4} + A^5x^5 \leq \mathbf{b}, \;\; x^i \in \mathbb{Z}^{n_i}_+$$

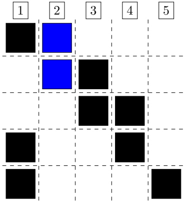
- Let this solutions be: $(\mathbf{0}, \bar{x}^2, \mathbf{0}, \mathbf{0}, \mathbf{0})$



Key idea - I

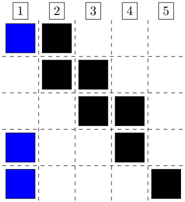
- Is
$$(\bar{x}^1, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) + (\mathbf{0}, \bar{x}^2, \mathbf{0}, \mathbf{0}, \mathbf{0})$$
guaranteed to be a feasible solution?

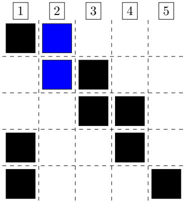




Key idea - I

- Is
$$(\bar{x}^1, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) + (\mathbf{0}, \bar{x}^2, \mathbf{0}, \mathbf{0}, \mathbf{0})$$
guaranteed to be a feasible solution?





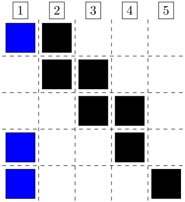
- **No!**

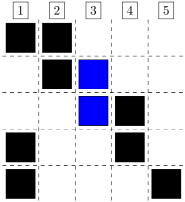
Key idea - I

- Is

$$(\bar{x}^1, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) + (\mathbf{0}, \mathbf{0}, \bar{x}^3, \mathbf{0}, \mathbf{0})$$

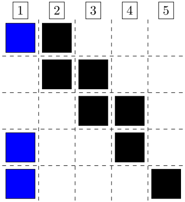
guaranteed to be a feasible solution?

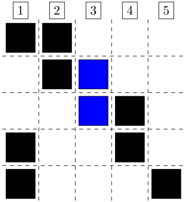




Key idea - I

- Is
$$(\bar{x}^1, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) + (\mathbf{0}, \mathbf{0}, \bar{x}^3, \mathbf{0}, \mathbf{0})$$
guaranteed to be a feasible solution?

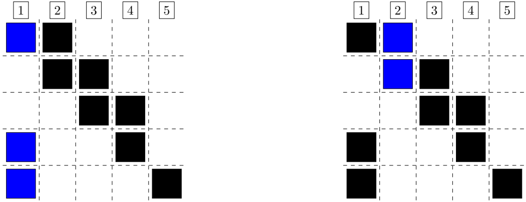


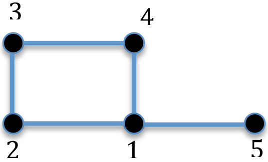


- **Yes!**

Key idea - I

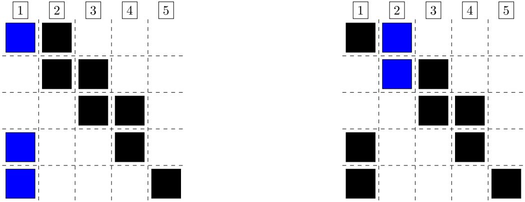
A graph theoretic perspective

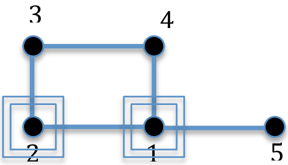




Key idea - I

A graph theoretic perspective

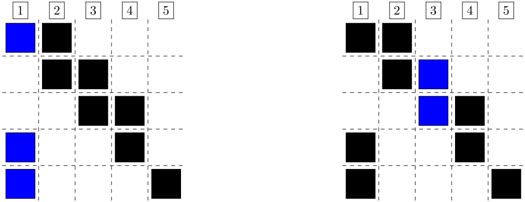


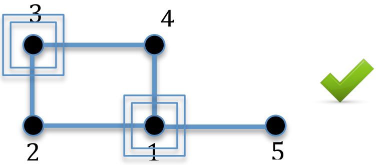


$\{1, 2\}$ is not a stable set in the graph!

Key idea - I

A graph theoretic perspective

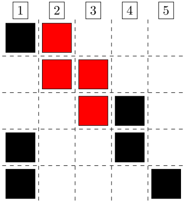




$\{1, 3\}$ is a stable set in the graph!

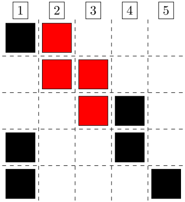
Key idea - II

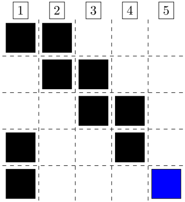
Generalizing the notion of stable sets



Key idea - II

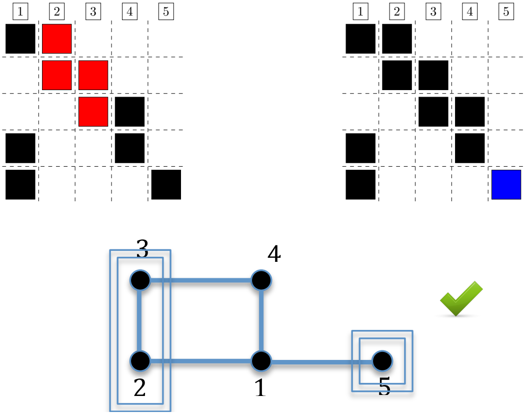
Generalizing the notion of stable sets





Key idea - II

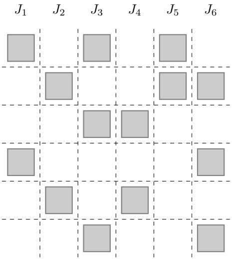
Generalizing the notion of stable sets



{{2, 3}, {5}} is a **general**-stable set in the graph!

2.2
Some definitions

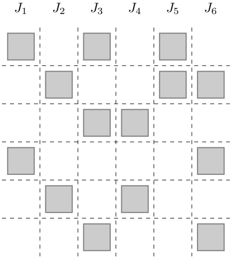
Describing sparsity of A

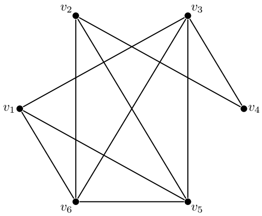


The matrix A with:

1. Column partition $\mathcal{J} := \{J_1, \dots, J_6\}$.
2. Unshaded boxes correspond to zeros in A.
3. Shaded boxes may have non-zero entries.

Describing sparsity of A





The matrix A with:

1. Column partition $\mathcal{J} := \{J_1, ..., J_6\}$.
2. Unshaded boxes correspond to zeros in A.
3. Shaded boxes may have non-zero entries.

The corresponding graph $G_{A,\mathcal{J}}$:

1. One node for every block of variables.
2. $(v_i, v_j) \in E$ if and only if there is a row in A with non-zero entries in both parts J_i and J_j.

Some graph-theoretic definition I: General-stable set

Definition (General stable set)

Let $G = (V, E)$ be a simple graph. Let $\mathcal{V}$ be a collection of subsets of the vertices V. We call a collection of subsets of vertices $M \subseteq 2^V$ a *general stable set subordinate to* $\mathcal{V}$ if the following hold:

1. Every set in M is contained in a set in $\mathcal{V}$.
2. The sets in M are pairwise disjoint.
3. There are no edges of G with endpoints in distinct sets in M.

Some graph-theoretic definition I: General-stable set

Definition (General stable set)

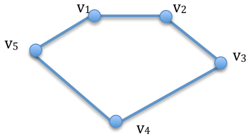
Let $G = (V, E)$ be a simple graph. Let $\mathcal{V}$ be a collection of subsets of the vertices V. We call a collection of subsets of vertices $M \subseteq 2^V$ a *general stable set subordinate to* $\mathcal{V}$ if the following hold:

1. Every set in M is contained in a set in $\mathcal{V}$.
2. The sets in M are pairwise disjoint.
3. There are no edges of G with endpoints in distinct sets in M.

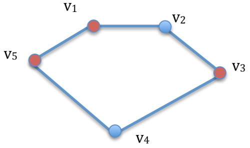
Example:

1. $\mathcal{V} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_1, v_5\}\}$
2. $\mathcal{M} = \{\{v_3\}, \{v_1, v_5\}\}$

Original Graph:



General Stable Set:



Some graph-theoretic definition II: General-chromatic number

Consider a simple graph $G = (V, E)$ and a collection $\mathcal{V}$ of subset of vertices.

- *General-chromatic number with respect to $\mathcal{V}$* (Denoted as $\bar{\eta}^{\mathcal{V}}_{(G)}$): It is the smallest number of general-stable sets $M^1, \dots, M^k$ subordinate to $\mathcal{V}$ that cover all vertices of the graph (that is, every vertex $v \in V$ belongs to a set in one of the $\mathcal{M}^i$'s).

Some graph-theoretic definition II: General-chromatic number

Consider a simple graph $G = (V, E)$ and a collection $\mathcal{V}$ of subset of vertices.

- *General-chromatic number with respect to $\mathcal{V}$* (Denoted as $\bar{\eta}^{\mathcal{V}}_{(G)}$): It is the smallest number of general-stable sets $M^1, \dots, M^k$ subordinate to $\mathcal{V}$ that cover all vertices of the graph (that is, every vertex $v \in V$ belongs to a set in one of the $\mathcal{M}^i$'s).
- *Fractional general-chromatic number with respect to $\mathcal{V}$* (Denoted as $\eta^{\mathcal{V}}_{(G)}$): Given a general stable set M subordinate to $\mathcal{V}$, let $\chi_M \in \{0,1\}^{|V|}$ denote its incidence vector (that is, for each vertex $v \in V$, $\chi_M(v) = 1$ if v belongs to a set in M, and $\chi_M(v) = 0$ otherwise.) Then we define the *fractional general-chromatic number*

$$\begin{aligned} \eta^{\mathcal{V}(G)} = \; & \min \sum_M y_M \\ & \text{s.t.} \sum_M y_M \chi_M \geq \mathbf{1} \\ & y_M \geq 0 \quad \forall \mathcal{M}, \end{aligned} \tag{2}$$

where the summations range over all general stable sets subordinate to $\mathcal{V}$.

2.3
Algorithm

Algorithm

- **Input:** Interaction graph $G_{A,\mathcal{J}} = (V, E)$; A collection $\mathcal{V}$ of subset of vertices. Let $\mathcal{M}$ be the collection of all general stable sets wrt $\mathcal{V}$.

Algorithm

- **Input:** Interaction graph $G_{A,\mathcal{J}} = (V, E)$; A collection $\mathcal{V}$ of subset of vertices. Let $\mathcal{M}$ be the collection of all general stable sets wrt $\mathcal{V}$.
- **Solve** Subproblems: For $W \subseteq U \in \mathcal{V}$.

$$\begin{aligned} w^W := \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j = 0, \forall \text{ variables not in } W. \\ & x_j \in \mathbb{Z}_{++} \end{aligned}$$

Optimal solution: x^W

Algorithm

- **Input:** Interaction graph $G_{A,\mathcal{J}} = (V, E)$; A collection $\mathcal{V}$ of subset of vertices. Let $\mathcal{M}$ be the collection of all general stable sets wrt $\mathcal{V}$.
- **Solve** Subproblems: For $W \subseteq U \in \mathcal{V}$.

$$\begin{aligned} w^W := \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j = 0, \forall \text{ variables } \textcolor{red}{\text{not in }} W. \\ & x_j \in \mathbb{Z}_{++} \end{aligned}$$

Optimal solution: x^W

- **Patching** Integer program:

$$\begin{aligned} z^A := \max \quad & \textstyle\sum_{\hat{U}} w^W y_W \\ \text{s.t.} \quad & y_{W_1} + y_{W_2} \leq 1 \text{ if } W_1 \cap W_2 \neq \emptyset \\ & y_{W_1} + y_{W_2} \leq 1 \text{ if } v_i \in W_i, (v_1, v_2) \in E \\ y \in \{0,1\}^{|\mathcal{W}|} \end{aligned}$$

Optimal solution: y^*

Algorithm

- **Input:** Interaction graph $G_{A,\mathcal{J}} = (V, E)$; A collection $\mathcal{V}$ of subset of vertices. Let $\mathcal{M}$ be the collection of all general stable sets wrt $\mathcal{V}$.
- **Solve** Subproblems: For $W \subseteq U \in \mathcal{V}$.

$$\begin{aligned} w^W := \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j = 0, \forall \text{ variables not in } W. \\ & x_j \in \mathbb{Z}_{++} \end{aligned}$$

  Optimal solution: x^W

- **Patching** Integer program:

$$\begin{aligned} z^A := \max \quad & \textstyle\sum_{\hat{U}} w^W y_W \\ \text{s.t.} \quad & y_{W_1} + y_{W_2} \leq 1 \text{ if } W_1 \cap W_2 \neq \emptyset \\ & y_{W_1} + y_{W_2} \leq 1 \text{ if } v_i \in W_i, (v_1, v_2) \in E \\ & y \in \{0,1\}^{|\mathcal{W}|} \end{aligned}$$

  Optimal solution: y^*

- **Output:** $x^A := \sum_{M \in \mathcal{W}} y^*_W x^W$

Performance guarantee

Theorem

Let x^* be the optimal solution of packing IP, and x^A be the solution produced by Algorithm, with column partition $\mathcal{J}$ (resulting in packing interaction graph $G_{A,\mathcal{J}}$) and list $\mathcal{V}$. Then

$$c^\top x^A \geq \left(\frac{1}{\eta^{\mathcal{V}}(G_{A,\mathcal{J}})}\right) \cdot c^\top x^*.$$

2.4
Computations

Two-stage instances

Table: Gaps for two-stage instances

Instance Features	#Variables	Algo.	CPLEX	GRASP
nv1-5s100/z5	600	2.321(2)	NA(5)	3.375
nv1-5s100/z10	600	15.494	21.904	19.365
nv1-5s100/z15	600	15.278	25.034	18.942
nv1-5s100/z20	600	22.265	40.015	24.344
nv1-5s150/z5	900	11.300	12.747	14.857
nv1-5s150/z10	900	15.792	24.169	19.244
nv1-5s150/z15	900	17.835	34.535	19.753
nv1-5s150/z20	900	26.705	26.705	26.705
nv1-5s200/z5	1,200	13.932	16.999	17.849
nv1-5s200/z10	1,200	12.056	22.763	15.018
nv1-5s200/z15	1,200	21.282	41.441	25.029
nv1-5s200/z20	1,200	28.571	54.171	28.571

Three stage instances

Table: Gaps for three stage instances

Instance Features	#Variables	Algo.	CPLEX	GRASP
nv1-5-25s100/z5	3,100	18.67	4.22	18.95
nv1-5-25s100/z10	3,100	26.94	29.81	30.09
nv1-5-25s100/z15	3,100	39.50	51.89	43.18
nv1-5-25s100/z20	3,100	47.62	68.97	51.64
nv1-5-25s150/z5	4,650	24.07	19.41	24.88
nv1-5-25s150/z10	4,650	39.22	47.07	44.82
nv1-5-25s150/z15	4,650	53.61	100.00	58.21
nv1-5-25s150/z20	4,650	59.38	100.00	60.14
nv1-5-25s200/z5	6,200	28.74	26.17	32.76
nv1-5-25s200/z10	6,200	47.83	56.69	53.17
nv1-5-25s200/z15	6,200	58.97	100.00	62.88
nv1-5-25s200/z20	6,200	60.59	100.00	60.88

Random Graph Instances

Table: Relative gap of various algorithms (random graph instances)

Instance Features	#Variables	Algo.	CPLEX	GRASP
nv50pb3s400/z5	20,000	9.184	100.000	16.501
nv50pb3s400/z10	20,000	7.824	100.000	16.006
nv50pb3s400/z15	20,000	7.855(1)	10.131	13.105(1)
nv50pb3s400/z20	20,000	NA(5)	100.000	NA(5)
nv50pb5s400/z5	20,000	12.935	100.000	18.427
nv50pb5s400/z10	20,000	14.985	100.000	19.135
nv50pb5s400/z15	20,000	18.642	100.000	25.027
nv50pb5s400/z20	20,000	NA(5)	100.000	3.700(1)
nv50pb8s400/z5	20,000	13.278	100.000	18.241
nv50pb8s400/z10	20,000	16.043	100.000	19.998
nv50pb8s400/z15	20,000	22.924	100.000	29.355
nv50pb8s400/z20	20,000	1.961	100.000	2.480
nv50pb10s400/z5	20,000	25.434	100.000	28.343
nv50pb10s400/z10	20,000	32.628	100.000	33.007
nv50pb10s400/z15	20,000	34.253	100.000	39.144
nv50pb10s400/z20	20,000	8.000	100.000	8.676

Random Graph Instances - II

Table: Relative gap of various algorithms (random graph instances) - II

Instance Features	#Variables	Gap^{A}	Gap^{IP}	Gap_{GS}^{GS}
nv100pb3s200/z5	20,000	14.832	100.000	19.134
nv100pb3s200/z10	20,000	15.779	100.000	22.794
nv100pb3s200/z15	20,000	22.779	100.000	26.653
nv100pb3s200/z20	20,000	19.763	100.000	25.879
nv100pb5s200/z5	20,000	23.966	100.000	26.188
nv100pb5s200/z10	20,000	27.320	100.000	29.188
nv100pb5s200/z15	20,000	37.675	100.000	38.463
nv100pb5s200/z20	20,000	30.815	100.000	34.126
nv100pb8s200/z5	20,000	35.643	100.000	35.248
nv100pb8s200/z10	20,000	46.136	100.000	41.337
nv100pb8s200/z15	20,000	53.383	100.000	52.565
nv100pb8s200/z20	20,000	42.604	100.000	48.049
nv100pb10s200/z5	20,000	42.259	100.000	38.650
nv100pb10s200/z10	20,000	56.557	100.000	49.064
nv100pb10s200/z15	20,000	60.890	100.000	61.593
nv100pb10s200/z20	20,000	50.665	100.000	55.050

Conclusion

Some comments:

- Our approximation algorithm almost always obtains a significantly better solution than `CPLEX` and `GRASP`.

Conclusion

Some comments:

- Our approximation algorithm almost always obtains a significantly better solution than `CPLEX` and `GRASP`.
- Our algorithm is very easily *parallelizable*. We may solve each of the sub-IPs separately.
- Dual bounds: Also *parallelizable:*

$$\begin{aligned} z^{UB} = \max \quad & c^T x \\ s.t. \quad & \sum_{j \in \text{ variables in } (U)} c_j x_j \leq \underbrace{w^U}_{Opt.obj.ofsubproblem}, \forall U \in \mathcal{V} \\ & \mathbf{0} \leq x \leq \mathbf{1} \end{aligned}$$

[Huchette, D., Vielma 16]