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
Using Sparsity to Design Primal Heuristics

Introduction

New randomization step for Feasibility Pump (FP)

Approximate algorithm for sparse packing problems

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:
Introduction

New randomization step for Feasibility Pump (FP)

Approximate algorithm for sparse packing problems

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\textsuperscript{1}, Marco Molinaro\textsuperscript{2}, and Domenico Salvagnin\textsuperscript{3}

\textsuperscript{1}Revenue Analytics, USA
\textsuperscript{2}PUC-Rio, Brazil
\textsuperscript{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 $y$)
- **Input**: 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, $\bar{x} = \ell_1$-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 $\sum |x_i - \tilde{x}_i|$ (we say, $\bar{x} = \ell_1$-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 ($\ell_1$-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.
Propose a modification of WalkSAT for the **mixed-binary** case.
- Show that this modified algorithm "works well" on mixed-binary instances that are **decomposable**.

**Analyse 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

  $$\frac{1}{s}$$

  $s$ # 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 differ from $x^*$.

- In each step, with probability at least

  $$\frac{1}{s}$$

  # non-zeros in violated constraint

  we choose to flip coordinate **where they differ**

  a positive constant

- With probability at least $\frac{1}{s}$, 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 $\sim k^{2^n/k}$ 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 \(\{(x, y) \mid Ax + By \leq b\}\) (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 \(\lambda\).

    (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
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

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

\[ P_i^l = P_i \cap (\{0, 1\}^{n_i} \times \mathbb{R}^{d_i}), \]

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

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(l) 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:

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

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

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

Note: Naive Feasibility Pump with original randomization may fail to converge for these instances.
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 $\tilde{x} \in \{0, 1\}^n$: $\text{AltProj}(\tilde{x}) := \text{round}(\ell_1\text{-proj}(\tilde{x}))$. 

Proposition (Length of cycle)

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

Proposition (Worst-case stabilization time)

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

# of iterations FPW $\leq \lceil \text{# iterations of } \text{AltProj}^* \rceil$

Number of stallings $\times \max_{\tilde{x} \in \{0, 1\}^n} \min_{k \in \mathbb{Z}^+} \text{altProj}_k(\tilde{x}) = \text{AltProj}^*(\tilde{x})$

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

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

\[
\text{# of iterations FPW} \leq \left[ \text{# iterations of AltProj}^* \right] \times \\
\text{Number of stallings}
\]

\[
\max_{\tilde{x} \in \{0, 1\}^n} \min\{k : \text{altProj}^k(\tilde{x}) = \text{altProj}^*(\tilde{x})\}.
\]

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

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

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

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

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

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

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 \( \text{AltProj}^* \) step.
Proof sketch - II

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

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 \( \bar{x} \in \{0, 1\}^n \) not in \( \text{Proj}_x(P) \cap \{0, 1\}^n \). Suppose \( \text{altProj}(\bar{x}) = \bar{x} \).
Proof sketch - II

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

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

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

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 \(\|\text{altProj}^*(\tilde{x}') - x^*\|_0 \leq \|\tilde{x} - x^*\|_0 - 1\).*
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 \( T_T \) (uniformly from \{10, \ldots, 30\}).
- Flip the \( \min\{F, T_T\} \) variables with highest fractionality.

New randomization

- Flip the \( \min\{F, T_T\} \) variables with highest fractionality.
- If \( F < T_T \), then:
  - let \( S \) be the union of the supports of the constraints that are not satisfied by the current point \((\bar{x}, \bar{y})\).
  - Select uniformly at random \( \min\{|S|, T_T - |F|\} \) 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

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

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

<table>
<thead>
<tr>
<th>seed</th>
<th># found</th>
<th>itr.</th>
<th>time (s)</th>
<th>% gap</th>
<th>% modified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>279</td>
<td>43</td>
<td>8.24</td>
<td>48%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>279</td>
<td>44</td>
<td>8.40</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>277</td>
<td>43</td>
<td>8.32</td>
<td>48%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>280</td>
<td>42</td>
<td>8.07</td>
<td>48%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>276</td>
<td>42</td>
<td>8.26</td>
<td>51%</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>277</td>
<td>43</td>
<td>8.29</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>278</td>
<td>43</td>
<td>8.17</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>273</td>
<td>43</td>
<td>8.16</td>
<td>49%</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>282</td>
<td>42</td>
<td>8.13</td>
<td>49%</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>278</td>
<td>42</td>
<td>8.33</td>
<td>50%</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>278</td>
<td>43</td>
<td>8.24</td>
<td>49%</td>
</tr>
</tbody>
</table>

**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!
Story 2: Approximate algorithm for sparse packing problems
Joint work with: Qianyi Wang\textsuperscript{4} and Marco Molinaro\textsuperscript{5}

\textsuperscript{4}Uber Inc.
\textsuperscript{5}PUC-Rio, Brazil
Packing instances

\[
\max \quad c^\top x \\
\text{s.t.} \quad Ax \leq b \\
x \in \mathbb{Z}^n_+, \\
\]

where \( A \) and \( b \) is non-negative.
Using Sparsity to Design Primal Heuristics

Introduction

New randomization step for Feasibility Pump (FP)

Approximate algorithm for sparse packing problems

Key ideas

Some definitions

Algorithm and main result

Computations

Packing instances

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

where $A$ and $b$ is non-negative.

High-level sketch of algorithm

The algorithm runs in two phases.
Packing instances

\[
\begin{align*}
\max & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad x \in \mathbb{Z}_+^n,
\end{align*}
\]

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

\[ \begin{align*}
\text{max} & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad x \in \mathbb{Z}^n_+, 
\end{align*} \]

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 $A$.

\[
\begin{align*}
A^1 x^1 + A^2 x^2 + A^3 x^3 + Ax^4 + A^5 x^5 & \leq b, \ x^i \in \mathbb{Z}^n_i
\end{align*}
\]
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.

\[
A^1 x^1 + A^2 x^2 + A^3 x^3 + A x^4 + A^5 x^5 \leq b, \ x^i \in \mathbb{Z}^n_i
\]

- Let this solution be: \((\bar{x}^1, 0, 0, 0, 0)\)
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.

\[ A^1 x^1 + A^2 x^2 + A^3 x^3 + A^4 x^4 + A^5 x^5 \leq b, \ x^i \in \mathbb{Z}^n_i \]

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

![Diagram of variable blocks]
Key idea - 1

Is

\[(\tilde{x}^1, 0, 0, 0, 0) + (0, \tilde{x}^2, 0, 0, 0)\]

guaranteed to be a feasible solution?
Key idea - I

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

- No!
Key idea - I

Is

\[ (\bar{x}^1, 0, 0, 0, 0) + (0, 0, \bar{x}^3, 0, 0) \]

guaranteed to be a feasible solution?
Key idea - I

Is

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

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 - 1

A graph theoretic perspective

\{1, 3\} is a stable set in the graph!
Key idea - II

Generalizing the notion of stable sets

1 2 3 4 5

{2, 3}, {5} is a general-stable set in the graph!
Using Sparsity to Design Primal Heuristics

Introduction
New randomization step for Feasibility Pump (FP)
Approximate algorithm for sparse packing problems

Key ideas
Some definitions
Algorithm and main result
Computations

Key idea - II

Generalizing the notion of stable sets
Key idea - II

Generalizing the notion of stable sets

\{\{2, 3\}, \{5\}\} is a \textbf{general}-stable set in the graph!
2.2

Some definitions
Describing sparsity of $A$

The matrix $A$ with:

1. **Column partition**
   \[ \mathcal{J} := \{ J_1, \ldots, J_6 \}. \]

2. **Unshaded boxes**
   correspond to *zeros* in $A$.

3. **Shaded boxes** may have non-zero entries.
The matrix $A$ with:

1. Column partition $\mathcal{J} := \{J_1, \ldots, J_6\}$.
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 $J_j$. 

The matrix $A$ with:

\begin{align*}
J_1 & | J_2 & J_3 & J_4 & J_5 & J_6 \\
\begin{array}{cccccc}
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\end{array}
\end{align*}

\begin{align*}
G_{A, \mathcal{J}}: \\
\begin{array}{cccccc}
v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\
\begin{array}{cccccc}
v_1 & & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & (v_i, v_j) \in E & & \\
\begin{array}{cccccc}
& & & & & \\
\end{array} & & & & & \\
\end{array}
\end{align*}
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. $V = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_1, v_5\}\}$
2. $M = \{\{v_3\}, \{v_1, v_5\}\}$

Original Graph:

```
  v1  v2  v4  v3  v5
```

General Stable Set:

```
  v1  v2  v4  v3  v5
```

---

In the original graph, the vertices are connected as follows:
- $v_1$ is connected to $v_2$ and $v_5$.
- $v_2$ is connected to $v_3$.
- $v_3$ is connected to $v_4$.
- $v_4$ is connected to $v_5$. 

In the general stable set, the vertices $v_3$, $v_1$, and $v_5$ are included, forming a stable set where no two vertices are connected by an edge.
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 $\bar{\eta}_{(G)}^\mathcal{V}$): It is the smallest number of general-stable sets $M^1, \ldots, M^k$ 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 $M^i$'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 $\tilde{\eta}_{\mathcal{V}(G)}$): It is the smallest number of general-stable sets $M^1, \ldots, M^k$ 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 $M^i$'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

$$\eta_{\mathcal{V}(G)} = \min \sum_M y_M$$

s.t. $\sum_M y_M \chi_M \geq 1$ \hspace{1cm} (2)

$$y_M \geq 0 \hspace{1cm} \forall M,$$

where the summations range over all general stable sets subordinate to $\mathcal{V}$. 
2.3

Algorithm
Algorithm

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

**Algorithm**

1. Solve Subproblems: For $W \subseteq \cup \in \mathcal{V}$.
   - $w_W := \max c^T x \text{ s.t. } Ax \leq b, x_j = 0, \forall \text{ variables not in } W$.
   - Optimal solution: $x_W$

2. Patching Integer program:
   - $z_A := \max \sum \hat{U} w_W y_W \text{ s.t. } y_W 1 + y_W 2 \leq 1 \text{ if } W_1 \cap W_2 \neq \emptyset$
   - $y \in \{0, 1\} |W|
   - Optimal solution: $y^*$

3. Output:
   - $x_A := \sum_{M \in W} y^* W x_W$
Algorithm

- **Input:** Interaction graph $G_{A,J} = (V,E)$; 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 $W \subseteq U \in \mathcal{V}$.

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

Optimal solution: $x^W$
Algorithm

- **Input:** Interaction graph $G_{A,J} = (V, E)$; 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 $W \subseteq U \in \mathcal{V}$.

  \[
  w^W := \max_{c^\top x} \quad \text{s.t.} \quad Ax \leq b
  \]
  \[
  x_j = 0, \forall \text{ variables not in } W.
  \]
  \[
  x_j \in \mathbb{Z}_{++}
  \]

  Optimal solution: $x^W$

- **Patching** Integer program:

  \[
  z^A := \max \sum_{W} w^W y_W
  \]
  \[
  \text{s.t.} \quad y_{W_1} + y_{W_2} \leq 1 \text{ if } W_1 \cap W_2 \neq \emptyset
  \]
  \[
  y_{W_1} + y_{W_2} \leq 1 \text{ if } v_i \in W_i, (v_1, v_2) \in E
  \]

  $y \in \{0, 1\}^{|\mathcal{W}|}$

  Optimal solution: $y^*$
Algorithm

- **Input:** Interaction graph $G_{A,J} = (V, E)$; 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 $W \subseteq U \in \mathcal{V}$.

  $w^W := \max \quad c^T x$

  s.t. $Ax \leq b$

  $x_j = 0, \forall$ variables not in $W$.

  $x_j \in \mathbb{Z}_{++}$

Optimal solution: $x^W$

- **Patching** Integer program:

  $z^A := \max \quad \sum_{\hat{U}} w^W y_W$

  s.t. $y_{W_1} + y_{W_2} \leq 1$ if $W_1 \cap W_2 \neq \emptyset$

  $y_{W_1} + y_{W_2} \leq 1$ if $v_i \in W_i, (v_1, v_2) \in E$

  $y \in \{0, 1\}^{\lfloor |\mathcal{W}| \rfloor}$

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 $\nu$. Then

$$c^\top x^A \geq \left( \frac{1}{\eta^\nu(G_{A,\mathcal{J}})} \right) \cdot c^\top x^*.$$
2.4
Computations
## Two-stage instances

**Table:** Gaps for two-stage instances

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

**Table: Gaps for three stage instances**

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

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

<table>
<thead>
<tr>
<th>Instance Features</th>
<th>#Variables</th>
<th>Algo.</th>
<th>CPLEX</th>
<th>GRASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv50pb3s400/z5</td>
<td>20,000</td>
<td>9.184</td>
<td>100.00</td>
<td>16.501</td>
</tr>
<tr>
<td>nv50pb3s400/z10</td>
<td>20,000</td>
<td>7.824</td>
<td>100.00</td>
<td>16.006</td>
</tr>
<tr>
<td>nv50pb3s400/z15</td>
<td>20,000</td>
<td>7.855(1)</td>
<td>10.131</td>
<td>13.105(1)</td>
</tr>
<tr>
<td>nv50pb3s400/z20</td>
<td>20,000</td>
<td>NA(5)</td>
<td>100.00</td>
<td>NA(5)</td>
</tr>
<tr>
<td>nv50pb5s400/z5</td>
<td>20,000</td>
<td>12.935</td>
<td>100.00</td>
<td>18.427</td>
</tr>
<tr>
<td>nv50pb5s400/z10</td>
<td>20,000</td>
<td>14.985</td>
<td>100.00</td>
<td>19.135</td>
</tr>
<tr>
<td>nv50pb5s400/z15</td>
<td>20,000</td>
<td>18.642</td>
<td>100.00</td>
<td>25.027</td>
</tr>
<tr>
<td>nv50pb5s400/z20</td>
<td>20,000</td>
<td>NA(5)</td>
<td>100.00</td>
<td>3.700(1)</td>
</tr>
<tr>
<td>nv50pb8s400/z5</td>
<td>20,000</td>
<td>13.278</td>
<td>100.00</td>
<td>18.241</td>
</tr>
<tr>
<td>nv50pb8s400/z10</td>
<td>20,000</td>
<td>16.043</td>
<td>100.00</td>
<td>19.998</td>
</tr>
<tr>
<td>nv50pb8s400/z15</td>
<td>20,000</td>
<td>22.924</td>
<td>100.00</td>
<td>29.355</td>
</tr>
<tr>
<td>nv50pb8s400/z20</td>
<td>20,000</td>
<td>1.961</td>
<td>100.00</td>
<td>2.480</td>
</tr>
<tr>
<td>nv50pb10s400/z5</td>
<td>20,000</td>
<td>25.434</td>
<td>100.00</td>
<td>28.343</td>
</tr>
<tr>
<td>nv50pb10s400/z10</td>
<td>20,000</td>
<td>32.628</td>
<td>100.00</td>
<td>33.007</td>
</tr>
<tr>
<td>nv50pb10s400/z15</td>
<td>20,000</td>
<td>34.253</td>
<td>100.00</td>
<td>39.144</td>
</tr>
<tr>
<td>nv50pb10s400/z20</td>
<td>20,000</td>
<td>8.000</td>
<td>100.00</td>
<td>8.676</td>
</tr>
</tbody>
</table>
Random Graph Instances - II

<table>
<thead>
<tr>
<th>Instance Features</th>
<th>#Variables</th>
<th>Gap&lt;sup&gt;A&lt;/sup&gt;</th>
<th>Gap&lt;sup&gt;IP&lt;/sup&gt;</th>
<th>Gap&lt;sup&gt;GS&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv100pb3s200/z5</td>
<td>20,000</td>
<td>14.832</td>
<td>100.000</td>
<td>19.134</td>
</tr>
<tr>
<td>nv100pb3s200/z10</td>
<td>20,000</td>
<td>15.779</td>
<td>100.000</td>
<td>22.794</td>
</tr>
<tr>
<td>nv100pb3s200/z15</td>
<td>20,000</td>
<td>22.779</td>
<td>100.000</td>
<td>26.653</td>
</tr>
<tr>
<td>nv100pb3s200/z20</td>
<td>20,000</td>
<td>19.763</td>
<td>100.000</td>
<td>25.879</td>
</tr>
<tr>
<td>nv100pb5s200/z5</td>
<td>20,000</td>
<td>23.966</td>
<td>100.000</td>
<td>26.188</td>
</tr>
<tr>
<td>nv100pb5s200/z10</td>
<td>20,000</td>
<td>27.320</td>
<td>100.000</td>
<td>29.188</td>
</tr>
<tr>
<td>nv100pb5s200/z15</td>
<td>20,000</td>
<td>37.675</td>
<td>100.000</td>
<td>38.463</td>
</tr>
<tr>
<td>nv100pb5s200/z20</td>
<td>20,000</td>
<td>30.815</td>
<td>100.000</td>
<td>34.126</td>
</tr>
<tr>
<td>nv100pb8s200/z5</td>
<td>20,000</td>
<td>35.643</td>
<td>100.000</td>
<td>35.248</td>
</tr>
<tr>
<td>nv100pb8s200/z10</td>
<td>20,000</td>
<td>46.136</td>
<td>100.000</td>
<td>41.337</td>
</tr>
<tr>
<td>nv100pb8s200/z15</td>
<td>20,000</td>
<td>53.383</td>
<td>100.000</td>
<td>52.565</td>
</tr>
<tr>
<td>nv100pb8s200/z20</td>
<td>20,000</td>
<td>42.604</td>
<td>100.000</td>
<td>48.049</td>
</tr>
<tr>
<td>nv100pb10s200/z5</td>
<td>20,000</td>
<td>42.259</td>
<td>100.000</td>
<td>38.650</td>
</tr>
<tr>
<td>nv100pb10s200/z10</td>
<td>20,000</td>
<td>56.557</td>
<td>100.000</td>
<td>49.064</td>
</tr>
<tr>
<td>nv100pb10s200/z15</td>
<td>20,000</td>
<td>60.890</td>
<td>100.000</td>
<td>61.593</td>
</tr>
<tr>
<td>nv100pb10s200/z20</td>
<td>20,000</td>
<td>50.665</td>
<td>100.000</td>
<td>55.050</td>
</tr>
</tbody>
</table>
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**:

\[
    z^{UB} = \max \quad c^T x \\
    s.t. \quad \sum_{j \in \text{variables in (U)}} c_j x_j \leq w^U, \quad \forall U \in \mathcal{V} \\
    0 \leq x \leq 1
\]

[Huchette, D., Vielma 16]