Flooding Overcomes Small Covering Constraints

Christos Koufogiannakis and Neal E. Young
University of California, Riverside
Covering Integer Programs

\[ \min \ c \cdot x \]

subject to: \( Ax \geq b \)

\( x \leq u \)

\( x \in Z^n \)

Let \( \delta \) be the maximum number of variables per constraint (i.e. \( \delta = 2 \) for Vertex Cover).
Let $\delta$ be the maximum number of variables per constraint ($\delta = 2$ for this example).
Covering Integer Programs

\[ \begin{align*}
\min & \quad x_1 + 0.5x_2 + 6x_3 : \\
& 2x_1 + x_2 \geq 3 \\
& x_2 + 4x_3 \geq 5 \\
& x_2 \leq 2 \\
& x_1, x_2, x_3 \in \mathbb{Z}_+ 
\end{align*} \]

Let \( \delta \) be the maximum number of variables per constraint (\( \delta = 2 \) for this example).

Goal: Simple \( \delta \)-approximation algorithm, applicable in the sequential
Covering Integer Programs

\[
\begin{align*}
\min & \quad x_1 + 0.5x_2 + 6x_3 : \\
& 2x_1 + x_2 \geq 3 \\
& x_2 + 4x_3 \geq 5 \\
& x_2 \leq 2 \\
& x_1, x_2, x_3 \in \mathbb{Z}_+
\end{align*}
\]

Let \( \delta \) be the maximum number of variables per constraint \( (\delta = 2 \text{ for this example}) \).

Goal: Simple \( \delta \)-approximation algorithm, applicable in the sequential, \textit{distributed}
Covering Integer Programs

\[ \min \ x_1 + 0.5x_2 + 6x_3 : \]

\[ x_2 \leq 2 \]

\[ x_1, x_2, x_3 \in \mathbb{Z}_+ \]

Let \( \delta \) be the maximum number of variables per constraint \( (\delta = 2 \text{ for this example}) \).

**Goal:** Simple \( \delta \)-approximation algorithm, applicable in the sequential, distributed and *online* setting.
Covering Integer Programs

\[ \min \ x_1 + 0.5x_2 + 6x_3 : \]
\[ 2x_1 + x_2 \geq 3 \]
\[ x_2 \leq 2 \]
\[ x_1, x_2, x_3 \in Z_+ \]

Let \( \delta \) be the maximum number of variables per constraint (\( \delta = 2 \) for this example).

**Goal:** Simple \( \delta \)-approximation algorithm, applicable in the sequential, distributed and *online* setting.
Covering Integer Programs

\[ \min x_1 + 0.5x_2 + 6x_3 : \]
\[ 2x_1 + x_2 \geq 3 \]
\[ x_2 + 4x_3 \geq 5 \]
\[ x_2 \leq 2 \]
\[ x_1, x_2, x_3 \in \mathbb{Z}_+ \]

Let \( \delta \) be the maximum number of variables per constraint \( (\delta = 2 \text{ for this example}) \).

**Goal:** Simple \( \delta \)-approximation algorithm, applicable in the sequential, distributed and online setting.
Covering Integer Programs

\[
\min \ x_1 + 0.5x_2 + 6x_3 : \\
2x_1 + x_2 \geq 3 \\
x_2 + 4x_3 \geq 5 \\
x_2 \leq 2 \\
x_1, x_2, x_3 \in \mathbb{Z}_+
\]

Previous sequential \(\delta\)-approximation algorithms use the KC-inequalities and an ellipsoid-based method [Carr et al., 2000][Pritchard, 2009].
Covering Integer Programs

\[
\begin{align*}
\text{min } & \ x_1 + 0.5x_2 + 6x_3 : \\
& 2x_1 + x_2 \geq 3 \\
& x_2 + 4x_3 \geq 5 \\
& x_2 \leq 2 \\
& x_1, x_2, x_3 \in Z_+ 
\end{align*}
\]

Previous sequential \( \delta \)-approximation algorithms use the KC-inequalities and an ellipsoid-based method [Carr et al., 2000][Pritchard, 2009].

Slow, not suitable for the distributed or online setting.
Given a node-weighted graph, find a minimum-weight subset of the nodes, touching all the edges.

\[
\min c \cdot x : \\
x_u + x_v \geq 1 \quad \forall (u, v) \\
x \in \{0, 1\}^n
\]
Weighted Vertex Cover

Given a node-weighted graph, find a minimum-weight subset of the nodes, touching all the edges.

\[
\begin{align*}
\text{min } c \cdot x & : \\
x_u + x_v & \geq 1 \quad \forall (u, v) \\
x & \in \{0, 1\}^n
\end{align*}
\]
A sequential and online 2-approximation algorithm for Weighted VC

- “Edge discount”: reduce edge endpoints’ costs equally.
- Do edge discounts until zero-cost nodes form a cover. Return the cover formed by the zero-cost nodes.

[Bar-Yehuda and Even, 1981]
edge discount operation

Reduce both endpoints’ costs equally.
edge discount operation

Reduce both endpoints’ costs equally.
edge discount operation

Reduce both endpoints’ costs equally.
Theorem: $\text{cost}(C) \leq 2\text{OPT}$
Distributed Computation

- Proceed in rounds.
- In each round:
  - Each node exchanges $O(1)$ messages with immediate neighbors,
  - then does some computation.
- **goal:** Finish in a poly-log number of rounds.
Each round:

1. form independent rooted “stars”
2. coordinate discounts within stars

• Done when zero-cost vertices cover all edges.

goal: Done after $O(\log n)$ rounds ($n = \#\text{nodes}$).
How to form stars

1. Each node randomly chooses to be “boy” or “girl” (just for this round)
2. For the round, use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)†
3. Each boy chooses a random neighbor (girl of ≥ cost).

† In each round, every edge has a one in four chance of being used. (... will be used if low-cost endpoint is boy, high-cost endpoint is girl)
How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
3. Each boy chooses a random neighbor (girl of ≥ cost).
How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
3. Each boy chooses a random neighbor (girl of ≥ cost).
How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
3. Each boy chooses a random neighbor (girl of ≥ cost).
• How to form stars:
  1. Each node randomly chooses to be “boy” or “girl”.
  2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
  3. Each boy chooses a random neighbor (girl of ≥ cost).
How to form stars:

1. Each node randomly chooses to be “boy” or “girl”.
2. Use only edges from boys to higher-cost (or equal-cost) girls. (Pretend other edges don’t exist.)
3. Each boy chooses a random neighbor (girl of ≥ cost).
How girls allocate discounts

- Each girl allocates discounts greedily, in alphabetic order.

- If she *partially allocates* some boy’s discount, then...
  - with probability $1/2$:
    1. She revokes discounts to all other boys.
    2. She allocates full discount to that boy.

Some boys may be *jilted* (have no chance for discount).
Each girl allocates discounts greedily, in alphabetical order.

If she *partially allocates* some boy’s discount, then...

with probability 1/2:

She revokes discounts to all other boys. She allocates full discount to that boy.
Each girl allocates discounts greedily, in alphabetical order.

- If she *partially allocates* some boy’s discount, then...

  with probability 1/2:

  She revokes discounts to all other boys. She allocates full discount to that boy.
Each girl allocates discounts greedily, in alphabetical order.

If she *partially allocates* some boy’s discount, then...

with probability $1/2$:

She revokes discounts to all other boys. She allocates full discount to that boy.
Each girl allocates discounts greedily, in alphabetical order.

If she *partially allocates* some boy’s discount, then...

with probability 1/2:

She revokes discounts to all other boys. She allocates full discount to that boy.
each of girl’s boys is either:

- **jilted** (girl gives no chance of any discount)
- **not jilted** (girl gives at least 50% chance of discount)

\[ \text{not jilted.} \]
\[ 50\% \text{ chance of full discount.} \]

\[ \text{definitely jilted.} \]
\[ \text{no chance of any discount.} \]
Analysis

• Guaranteed to return a 2-approximate solution, since it implements the edge-discount algorithm.

• What about running time?

• **Goal**: Show $O(\log n)$ rounds (w.h.p.).
Analysis of number of rounds

- “Delete” edges when one endpoint’s cost becomes zero.

- **Lemma:** In each round, in expectation, a constant fraction of each boy’s active edges are deleted.

- **Proof:** (next)

**Corollary:** Number of rounds is $O(\log n^2) = O(\log n)$ in expectation and with high probability.
lemma: In each round, in expectation, a constant fraction of each boy’s active edges are deleted.

key observation:
girl would jilt boy $\Rightarrow$ her cost is going to zero regardless of what boy does.
girl would not jilt boy $\Rightarrow$ if boy chooses her, she has at least a 50% chance of zeroing boy’s cost...
key observation:
girl **would jilt** boy ⇒ her cost is going to zero regardless of what boy does.
girl **would not jilt** boy ⇒ if boy chooses her, she has at least a 50% chance of zeroing boy’s cost...

- **case (i):** At least half of boy’s girls **would jilt** him.
  ⇒ At least half of boy’s edges will be deleted regardless of what boy does.

- **case (ii):** At least half of boy’s girls **would not jilt** him.
  ⇒ Boy has at least a 50% chance of choosing a girl who has at least a 50% chance of zeroing his cost (deleting all his edges).
• There is a simple, fast, sequential, online and distributed 2-approximation algorithm for Weighted Vertex Cover!
More general Covering Problems

\[
\begin{align*}
\min & \quad x_u + 2x_v + 3x_w : \\
& x_u + x_v \geq 3 \\
& x_v + x_w \geq 4 \\
& x_u, x_v, x_w \in Z_+
\end{align*}
\]

Bar-Yehuda and Even’s algorithm does not extend to non 0/1 problems.
More general Covering Problems

\[
\begin{align*}
\min & \quad x_u + 2x_v + 3x_w : \\
& x_u + x_v \geq 3 \\
& x_v + x_w \geq 4 \\
& x_u, x_v, x_w \in Z_+
\end{align*}
\]

Bar-Yehuda and Even’s algorithm does not extend to non 0/1 problems.

Goal: Simple 2-approximation algorithm, applicable in the sequential, distributed and online setting.
Flooding algorithm

1. Let $x \leftarrow 0$.

2. While $\exists$ edge $(u, v)$ s.t. $\left\lfloor x_u \right\rfloor + \left\lfloor x_v \right\rfloor < b_{uv}$ do:

3. Raise $x_u$ at rate $1/c_u$ and $x_v$ at rate $1/c_v$ until $\left\lfloor x_u \right\rfloor + \left\lfloor x_v \right\rfloor \geq b_{uv}$.

4. Return $\left\lfloor x \right\rfloor$. 
Flooding algorithm

1. Let $x \leftarrow 0$.

2. While \( \exists \) edge \((u, v)\) s.t. \( x_u + x_v < b_{uv} \) do:

3. \hspace{0.5cm} Raise \( x_u \) at rate \( 1/c_u \) and \( x_v \) at rate \( 1/c_v \) until \( x_u + x_v \geq b_{uv} \).

4. Return \( \lfloor x \rfloor \).

\[
\begin{align*}
\min \ x_u + 2x_v + 3x_w : \\
x_u + x_v &\geq 3 \\
x_v + x_w &\geq 4 \\
x_u, x_v, x_w &\in \mathbb{Z}_+
\end{align*}
\]

\[
\begin{align*}
c_u &= 1 \\
c_v &= 2 \\
c_w &= 3
\end{align*}
\]

\[
\begin{align*}
x_u &= 0 \\
x_v &= 0 \\
x_w &= 0
\end{align*}
\]
Flooding algorithm

1. Let \( x \leftarrow 0 \).

2. While \( \exists \) edge \((u, v)\) s.t. \( x_u + x_v < b_{uv} \) do:

3. \( \) Raise \( x_u \) at rate \( 1/c_u \) and \( x_v \) at rate \( 1/c_v \) until \( x_u + x_v \geq b_{uv} \).

4. Return \( \lfloor x \rfloor \).

\[
\min x_u + 2x_v + 3x_w : \\
x_u + x_v \geq 3 \\
x_v + x_w \geq 4 \\
x_u, x_v, x_w \in \mathbb{Z}_+
\]

\[
\begin{align*}
c_u &= 1 \\
c_v &= 2 \\
c_w &= 3 \\
x_u &= 0 \\
x_v &= 0 \\
x_w &= 0
\end{align*}
\]
Flooding algorithm

1. Let \( x \leftarrow 0 \).
2. While \( \exists \) edge \((u, v)\) s.t. \( x_u + x_v < b_{uv} \) do:
3. \quad Raise \( x_u \) at rate \( 1/c_u \) and \( x_v \) at rate \( 1/c_v \) until \( x_u + x_v \geq b_{uv} \).
4. Return \( \lfloor x \rfloor \).

\[
\min x_u + 2x_v + 3x_w : \\
x_u + x_v \geq 3 \\
x_v + x_w \geq 4 \\
x_u, x_v, x_w \in \mathbb{Z}_+
\]

\[
c_u = 1 \quad c_v = 2 \quad c_w = 3
\]

\[
x_u = 2 \quad x_v = 1 \quad x_w = 0
\]
Flooding algorithm

1. Let \( x \leftarrow 0 \).
2. While \( \exists \) edge \((u, v)\) s.t. \( x_u + x_v < b_{uv} \) do:
3. \( \text{Raise } x_u \text{ at rate } 1/c_u \text{ and } x_v \text{ at rate } 1/c_v \text{ until } x_u + x_v \geq b_{uv}. \)
4. Return \( \lfloor x \rfloor \).

\[
\begin{align*}
\min x_u + 2x_v + 3x_w : \\
x_u + x_v &\geq 3 \\
x_v + x_w &\geq 4 \\
x_u, x_v, x_w &\in \mathbb{Z}_+
\end{align*}
\]

\[
\begin{align*}
c_u &= 1 & c_v &= 2 & c_w &= 3 \\
x_u &= 2 & x_v &= 1 & x_w &= 0
\end{align*}
\]
Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge $(u, v)$ s.t. $\left\lfloor x_u \right\rfloor + \left\lfloor x_v \right\rfloor < b_{uv}$ do:
3. Raise $x_u$ at rate $1/c_u$ and $x_v$ at rate $1/c_v$ until $\left\lfloor x_u \right\rfloor + \left\lfloor x_v \right\rfloor \geq b_{uv}$.
4. Return $\left\lfloor x \right\rfloor$.

\[
\begin{align*}
\text{min } & x_u + 2x_v + 3x_w : \\
& x_u + x_v \geq 3 \\
& x_v + x_w \geq 4 \\
& x_u, x_v, x_w \in \mathbb{Z}_+
\end{align*}
\]

\[
\begin{align*}
c_u &= 1 & c_v &= 2 & c_w &= 3 \\
x_u &= 2 & x_v &= 3 & x_w &= \frac{4}{3}
\end{align*}
\]
Flooding algorithm

1. Let \( x \leftarrow 0 \).
2. While \( \exists \) edge \((u, v)\) s.t. \( x_u + x_v < b_{uv} \) do:
3. \quad Raise \( x_u \) at rate \( 1/c_u \) and \( x_v \) at rate \( 1/c_v \) until \( x_u + x_v \geq b_{uv} \).
4. Return \( x \).

\[
\min x_u + 2x_v + 3x_w : \quad x_u + x_v \geq 3 \\
x_v + x_w \geq 4 \\
x_u, x_v, x_w \in \mathbb{Z}_+
\]

\[
c_u = 1 \quad c_v = 2 \quad c_w = 3
\]

\[
\begin{align*}
\text{Return} \quad & x_u = 2, x_v = 3, x_w = \left\lfloor \frac{4}{3} \right\rfloor = 1 \\
& \quad x_u = 2, x_v = 3, x_w = \left\lfloor \frac{4}{3} \right\rfloor = 1
\end{align*}
\]
Flooding algorithm

1. Let $x \leftarrow 0$.
2. While $\exists$ edge $(u, v)$ s.t. $\left[x_u\right] + \left[x_v\right] < b_{uv}$ do:
3. 
   Raise $x_u$ at rate $1/c_u$ and $x_v$ at rate $1/c_v$ until $\left[x_u\right] + \left[x_v\right] \geq b_{uv}$.
4. Return $\left[x\right]$.

Let $x^*$ be any feasible solution.

Each step starts with a non-yet-satisfied constraint, so $x_u^* > x_u$ or $x_v^* > x_v$.

Let $\text{residual}_c(x)$ be the min cost to increase $x$ to full feasibility.

The step increases the cost by $2\beta$ but it reduces $\text{residual}_c(x)$ by at least $\beta \iff 2$-approximation.
Results (sequential)

• **Covering Mixed Integer Programs:** Nearly linear time $\delta$-approximation algorithm. Improves over the previous ellipsoid-based (slow) algorithm.

• **Non-metric Facility Location:** Linear time $\delta$-approximation. $\delta$ is the maximum number of facilities that might serve a customer.

• **Covering problems with submodular cost function**
Results (online)

- **Online CMIP**: $\delta$-competitive algorithm.

- **Paging, Weighted Caching, File Caching, Connection Caching**: Generalize $k$-competitive algorithms i.e. Landlord, Harmonic, Greedy-Dual.

- **Upgradable Caching**: $(k+d)$-competitive algorithm, where $k$ is the cache size and $d$ is the number of upgradable parameters.
Results (distributed)

• Covering Mixed Integer Programs with 2 variables per constraint: 2-approximation in $O(\log |C|)$ rounds, where $|C|$ is the number of constraints. Also, 2-approximation in RNC.

2-approximation for Weighted Vertex Cover in $O(\log n)$ rounds.

• Covering Problems with at most $\delta$ variables per constraint: $\delta$-approximation in $O(\log^2 |C|)$ rounds.

PODC 2009
thank you