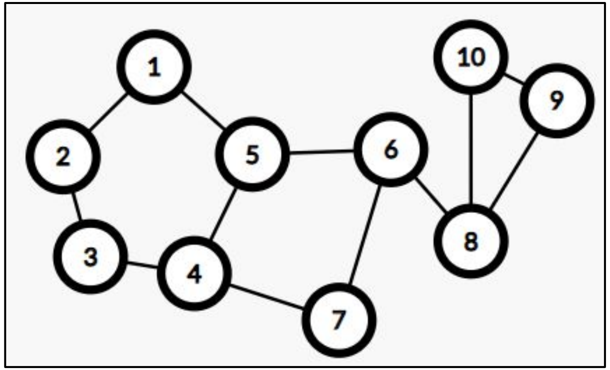
Feedback Vertex Set

Likith Rajesh, Sanchit Jindal, Virendra Kabra

FVS Definition

- Given undirected graph G(V,E) with non-negative vertex weights $w_i \geq 0 \quad \forall i \in V$
- Choose minimum-cost $S \subseteq V$ whose removal from the graph G[V] makes the remaining graph G[V-S] acyclic(forest)
- Input:G(V,E) Output:S($\subseteq$V)
- Example
  - S = {4, 8}



Linear Program

- Variables x_i for $\forall i \in V$
  - x_i is 1 iff vertex i is in chosen S
- $\mathcal{C}$ is the set of cycles C in G
- Relaxation: $x_i \geq 0$
- Issue: Exponential number of constraints
  - Example: Complete graph on n vertices has $O(2^n)$ cycles

Integer program for FVS (undirected graphs)

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in V} w_i x_i \\
\text{subject to} \quad & \sum_{i \in C} x_i \geq 1, && \forall C \in \mathcal{C}, \\
& x_i \in \{0, 1\}, && \forall i \in V.
\end{aligned}
$$

Dual

- Variables y_C for $\forall C \in \mathcal{C}$
  - One for each cycle
- Issue: Exponential number of variables
  - But only polynomial number of them will be non-zero in the algorithm

$$\begin{aligned} \text{maximize} \quad & \sum_{C \in \mathcal{C}} y_C \\ \text{subject to} \quad & \sum_{C \in \mathcal{C} : i \in C} y_C \leq w_i, \qquad \forall i \in V, \\ & y_C \geq 0, \qquad \forall C \in \mathcal{C}. \end{aligned}$$

- Complementary Slackness Conditions
  - First is tight
  - Second gives approximation factor

$$x_i > 0 \implies \sum_{C \in \mathcal{C} : i \in C} y_C = w_i$$

$$y_C > 0 \implies \sum_{i \in C} x_i = 1$$

Primal-Dual Algorithm

$y \leftarrow 0$
$S \leftarrow \emptyset$
while there exists a cycle C in G **do**
  Increase y_C until there is some $\ell \in C$ such that $\sum_{C' \in \mathcal{C} : \ell \in C'} y_{C'} = w_\ell$
  $S \leftarrow S \cup \{\ell\}$
  Remove ℓ from G
  Repeatedly remove vertices of degree one from G
return S

Analysis

- Algorithm ensures first Complementary Slackness condition is tight
  - For any i∈S, $\sum_{C\in\mathcal{C}:i\in C} y_C = w_i$
- So cost of ALG output is

$$\sum_{i\in S} w_i = \sum_{i\in S}\sum_{C:i\in C} y_C = \sum_{C\in\mathcal{C}} |S\cap C| y_C$$

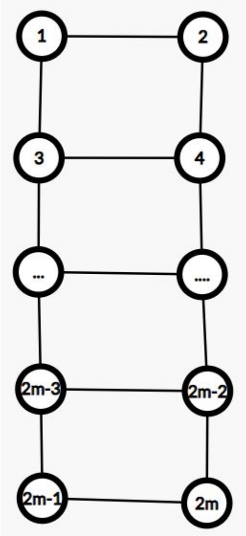
- If we bound |S∩C| ≤ α

$$\sum_{i\in S} w_i \leq \alpha \sum_{C\in\mathcal{C}} y_C \leq \alpha\cdot\mathrm{OPT}$$

- This is using Dual-feasible ≤ LP_OPT ≤ OPT

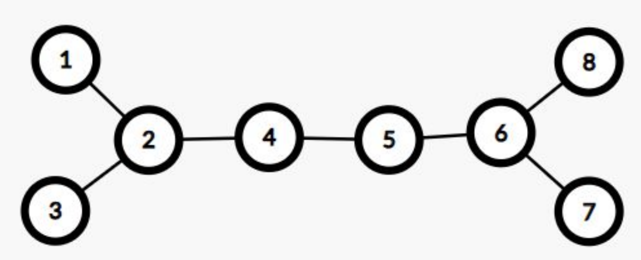
Analysis

- But choosing arbitrary cycles in algorithm-loop can lead to $|S \cap C| = O(n)$
  - Example: Ladder graph - $S = \{1, 3, \ldots, 2m-3\}$, C = outer cycle
  - Choosing smallest cycle also does not work
- Need to choose cycles more cleverly



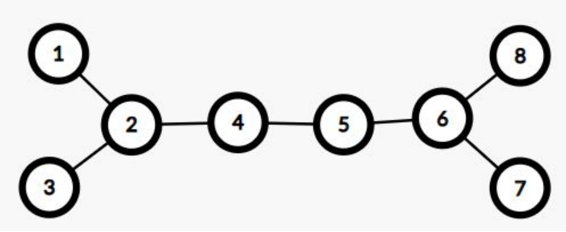
Improvements

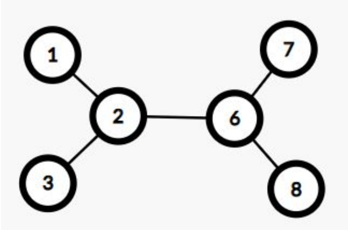
- Observation: For any path of degree-two vertices, at most one vertex is in S
  - Example: Remove 4 and algorithm removes degree-1 vertices



Improvements

- Lemma: In any graph that has no vertices of degree one, there is a cycle with at most $2\lceil \log_2 n \rceil$ vertices of degree three or more, and it can be found in linear time
  - Trivial if no vertices with degree≥3
  - Else
    - Compact all paths of degree-2 vertices
    - Run BFS on the modified graph
    - Every node in BFS tree has degree ≥ 3, so each layer has at least double the nodes of previous layer
    - So height ≤ $\lceil \log_2 n \rceil$ and cycle size ≤ 2 $\lceil \log_2 n \rceil$





Improved Algorithm

$y \leftarrow 0$
$S \leftarrow \emptyset$
Repeatedly remove vertices of degree one from G
while there exists a cycle in G **do**
    Find cycle C with at most $2\lceil \log_2 n \rceil$ vertices of degree three or more
    Increase y_C until there is some $\ell \in C$ such that $\sum_{C' \in \mathcal{C}: \ell \in C'} y_{C'} = w_\ell$
    $S \leftarrow S \cup \{\ell\}$
    Remove ℓ from G
    Repeatedly remove vertices of degree one from G
return S

Improved Analysis

- As before, we have

$$\sum_{i\in S} w_i = \sum_{i\in S}\sum_{C:i\in C} y_C = \sum_{C\in\mathcal{C}} |S\cap C| y_C$$

- Now, $y_C > 0$ only when C contains at most $2\lceil \log_2 n \rceil$ vertices with degree ≥ 3
- By the observation, each path of vertices of degree-2 joining vertices of degree ≥ 3 in C can contain at most one vertex of S
- So we get $y_C > 0 \Rightarrow |S \cap C| \leq 4\lceil \log_2 n \rceil$
- This gives a better approximation of $4\lceil \log_2 n \rceil$

Remarks

- This approximation factor is tight for this LP
  - Integrality gap is logarithmic
- Tighter approximation factor of 2 can be obtained by a different LP

- Applications
  - Operating Systems: Make dependency graph cycle-free for deadlock prevention

Thank You!