A Naive Algorithm for Feedback Vertex Set

Yixin Cao（操宜新）

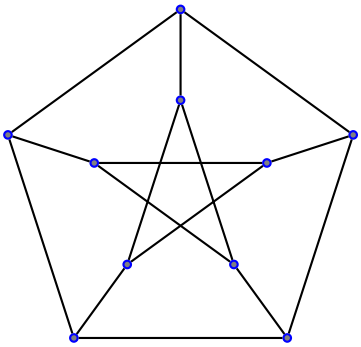
Department of Computing, Hong Kong Polytechnic University

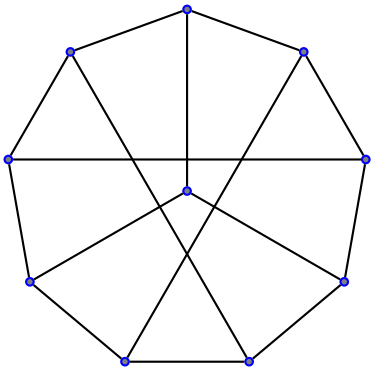
香港理工大學　電子計算學系

Symposium on Simplicity in Algorithms

New Orleans, LA January 10, 2018

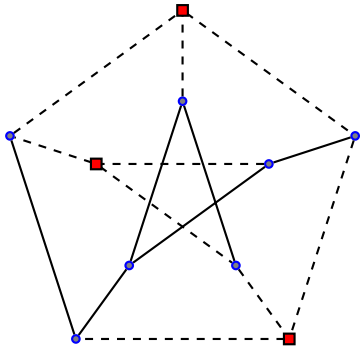
The problem

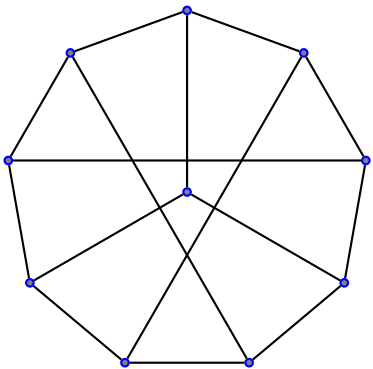




Given a graph G and an integer k, the feedback vertex set problem asks for the deletion of at most k vertices to make G a forest.

The problem





Given a graph G and an integer k, the feedback vertex set problem asks for the deletion of at most k vertices to make G a forest.

Origin

The constraint satisfaction problem (CSP) asks for the assignment of values to variables to satisfy a set of constraints.

[Freuder 1982]: can be solved in P-time when the constraint graph is a forest.

[Dechter Pearl 1987]: one way to solve the constraint satisfaction problem is to

1. find first a minimum feedback vertex set V_- of the constraint graph,
2. enumerate all possible assignments on them, and
3. then solve the remaining instance.

On an instance I on p variables, it takes $O(p^{|V_-|} \cdot |I|^{O(1)})$ time.

[Pearl 1988]: A similar application for Bayesian inference.

Origin

The constraint satisfaction problem (CSP) asks for the assignment of values to variables to satisfy a set of constraints.

[Freuder 1982]: can be solved in P-time when the constraint graph is a forest.

[Dechter Pearl 1987]: one way to solve the constraint satisfaction problem is to

1. find first a minimum feedback vertex set V_- of the constraint graph,
2. enumerate all possible assignments on them, and
3. then solve the remaining instance.

On an instance I on p variables, it takes $O(p^{|V_-|} \cdot |I|^{O(1)})$ time.

[Pearl 1988]: A similar application for Bayesian inference.

Origin

The constraint satisfaction problem (CSP) asks for the assignment of values to variables to satisfy a set of constraints.

[Freuder 1982]: can be solved in P-time when the constraint graph is a forest.

[Dechter Pearl 1987]: one way to solve the constraint satisfaction problem is to

1. find first a minimum feedback vertex set V_- of the constraint graph,
2. enumerate all possible assignments on them, and
3. then solve the remaining instance.

On an instance I on p variables, it takes $O(p^{|V_-|} \cdot |I|^{O(1)})$ time.

[Pearl 1988]: A similar application for Bayesian inference.

Origin

The constraint satisfaction problem (CSP) asks for the assignment of values to variables to satisfy a set of constraints.

[Freuder 1982]: can be solved in P-time when the constraint graph is a forest.

[Dechter Pearl 1987]: one way to solve the constraint satisfaction problem is to

1. find first a minimum feedback vertex set V_- of the constraint graph,
2. enumerate all possible assignments on them, and
3. then solve the remaining instance.

On an instance I on p variables, it takes $O(p^{|V_-|} \cdot |I|^{O(1)})$ time.

[Pearl 1988]: A similar application for Bayesian inference.

Origin

The constraint satisfaction problem (CSP) asks for the assignment of values to variables to satisfy a set of constraints.

[Freuder 1982]: can be solved in P-time when the constraint graph is a forest.

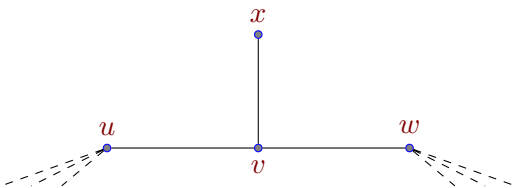
[Dechter Pearl 1987]: one way to solve the constraint satisfaction problem is to

1. find first a minimum feedback vertex set V_- of the constraint graph,
2. enumerate all possible assignments on them, and
3. then solve the remaining instance.

On an instance I on p variables, it takes $O(p^{|V_-|} \cdot |I|^{O(1)})$ time.

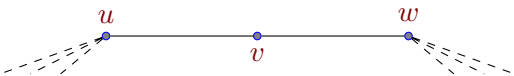
[Pearl 1988]: A similar application for Bayesian inference.

Low-degree vertices



A vertex of degree 1 can always be safely deleted.

Low-degree vertices



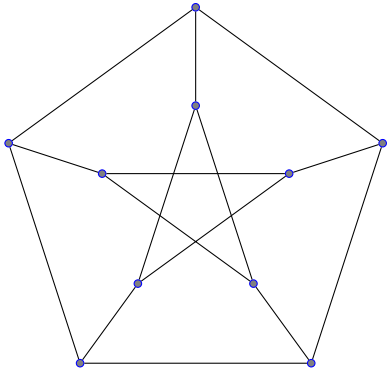
A degree-2 vertex in a solution can always be replaced with one of its neighbors.

Low-degree vertices

If u and w are not adjacent,
then we can delete v and add edge uw.

this operation is called "smoothen."

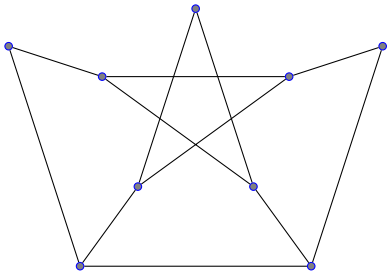
Low-degree vertices



If u and w are not adjacent,
then we can delete v and add edge uw.

this operation is called "smoothen."

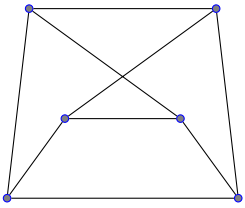
Low-degree vertices



If u and w are not adjacent,
then we can delete v and add edge uw.

this operation is called "smoothen."

Low-degree vertices



If u and w are not adjacent,
then we can delete v and add edge uw.

this operation is called "smoothen."


BACKLIGHTINGHOSTEL
逆光·小舍
兰家湾36号

Greed ... is good. Greed is right. Greed works.

Wall street

Algorithms

[Erdős Pósa 1962]: A graph of minimum degree ≥ 3 has a cycle of length $O(\log n)$.

- A trivial $O(\log n)$ approximation: find a shortest cycle, remove all vertices.
- A nontrivial $(\log k)^{O(k)}$-time parameterized algorithm: after a polynomial kernel.

Large-degree vertices

[Becker Geiger 1996] [Bafna Berman Fujito 1999]: A greedy 2-approximation (Local ratio)

[Chudak Goemans Hochbaum Williamson 1998]: A primal-dual interpretation

[Fujito 1996]: Two new primal-dual algorithms

Algorithms

Small-degree vertices

[Erdős Pósa 1962]: A graph of minimum degree ≥ 3 has a cycle of length $O(\log n)$.

- A trivial $O(\log n)$ approximation: find a shortest cycle, remove all vertices.
- A nontrivial $(\log k)^{O(k)}$-time parameterized algorithm: after a polynomial kernel.

Large-degree vertices

[Becker Geiger 1996] [Bafna Berman Fujito 1999]: A greedy 2-approximation (Local ratio)

[Chudak Goemans Hochbaum Williamson 1998]: A primal-dual interpretation

[Fujito 1996]: Two new primal-dual algorithms

Algorithms

Small-degree vertices

[Erdős Pósa 1962]: A graph of minimum degree ≥ 3 has a cycle of length $O(\log n)$.

- A trivial $O(\log n)$ approximation: find a shortest cycle, remove all vertices.
- A nontrivial $(\log k)^{O(k)}$-time parameterized algorithm: after a polynomial kernel.

Large-degree vertices

[Becker Geiger 1996] [Bafna Berman Fujito 1999]: A greedy 2-approximation (Local ratio)

[Chudak Goemans Hochbaum Williamson 1998]: A primal-dual interpretation

[Fujito 1996]: Two new primal-dual algorithms

Parameterized algorithms

Algorithms running in time $O(f(k) \cdot n^{O(1)})$, where k is a parameter (the solution size).

	$f(k)$
Downey Fellows 1992	$(2k+1)^k$
Bodlaender 1994	$17(k^4)!$
Raman et al. 2002	$\max\{12^k, (4\log k)^k\}$
Kanj et al. 2004	$(2\log k + 2\log\log k + 18)^k$
Raman et al. 2006	$(12\log k/\log\log k + 6)^k$
Dehne et al. 2005	10.6^k
Guo et al. 2006	37.7^k
Chen et al. 2008	5^k
C Chen Liu 2010	3.83^k
Kociumaka Pilipczuk 2014	3.62^k

Parameterized algorithms

Algorithms running in time $O(f(k) \cdot n^{O(1)})$, where k is a parameter (the solution size).

	$f(k)$
Downey Fellows 1992	$(2k+1)^k$
Bodlaender 1994	$17(k^4)!$
Raman et al. 2002	$\max\{12^k, (4\log k)^k\}$
Kanj et al. 2004	$(2\log k + 2\log\log k + 18)^k$
Raman et al. 2006	$(12\log k/\log\log k + 6)^k$
Dehne et al. 2005	10.6^k
Guo et al. 2006	37.7^k
Chen et al. 2008	5^k
C Chen Liu 2010	3.83^k
Kociumaka Pilipczuk 2014	3.62^k

Parameterized algorithms

Algorithms running in time $O(f(k) \cdot n^{O(1)})$, where k is a parameter (the solution size).

	$f(k)$
Downey Fellows 1992	$(2k+1)^k$
Bodlaender 1994	$17(k^4)!$
Raman et al. 2002	$\max\{12^k, (4\log k)^k\}$
Kanj et al. 2004	$(2\log k + 2\log\log k + 18)^k$
Raman et al. 2006	$(12\log k/\log\log k + 6)^k$
Dehne et al. 2005	10.6^k
Guo et al. 2006	37.7^k
Chen et al. 2008	5^k
C Chen Liu 2010	3.83^k
Kociumaka Pilipczuk 2014	3.62^k

All c^k algorithms use technique "iterative compression."

Context

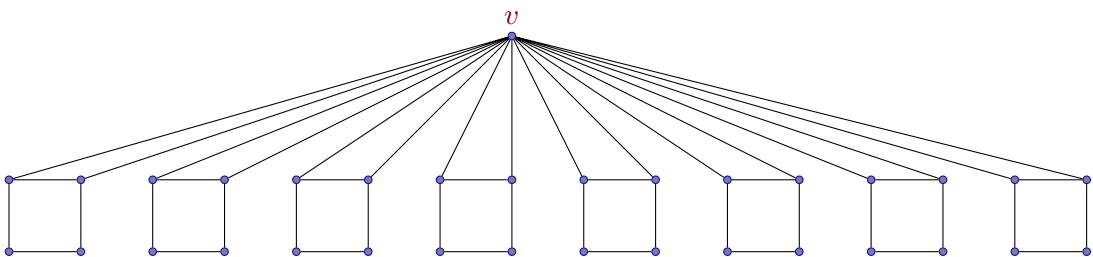
	vertex cover	feedback vertex set
to kill	edges	cycles
to make	independent set (edgeless)	forest (acyclic)
	treewidth 0	treewidth ≤ 1
approx	2	2
parameterized	1.2738^k	3.62^k

the *simplest nontrivial* vertex deletion problem

The Algorithm

A vertex of the largest degree is highly likely in the solution

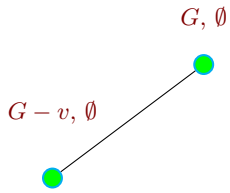
A vertex of the largest degree is highly likely in the solution, but *not* always.



A branching algorithm

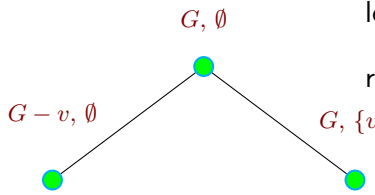
each node has two parts,
the graph G and the set F of *permanent* vertices.

A branching algorithm



left child: the vertex is put into the solution, hence deleted from G

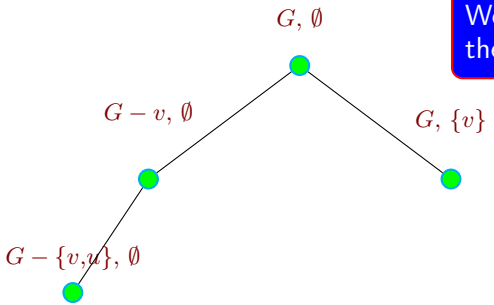
A branching algorithm



left child: the vertex is put into the solution, hence deleted from G

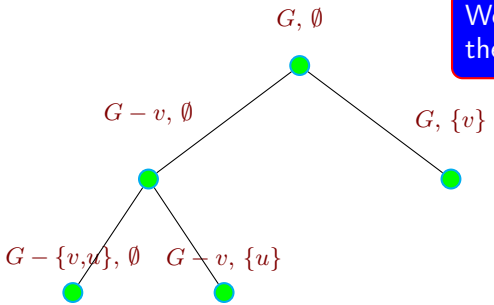
right child: the vertex is not in the solution, hence put into F

A branching algorithm



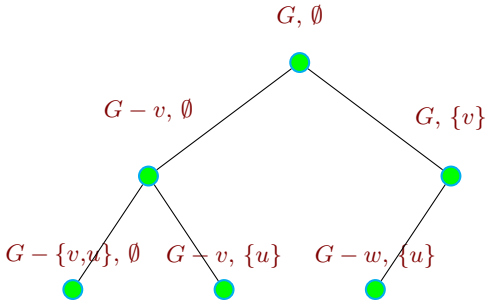
We always choose the vertex with the largest degree in $V(G) \setminus F$ to branch.

A branching algorithm

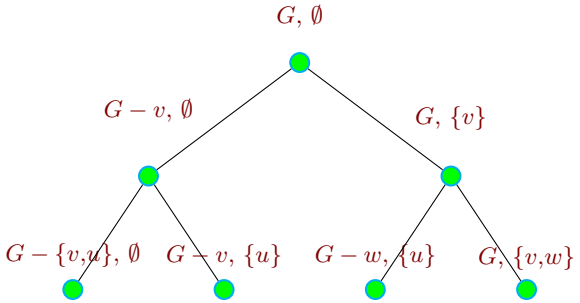


We always choose the vertex with the largest degree in $V(G) \setminus F$ to branch.

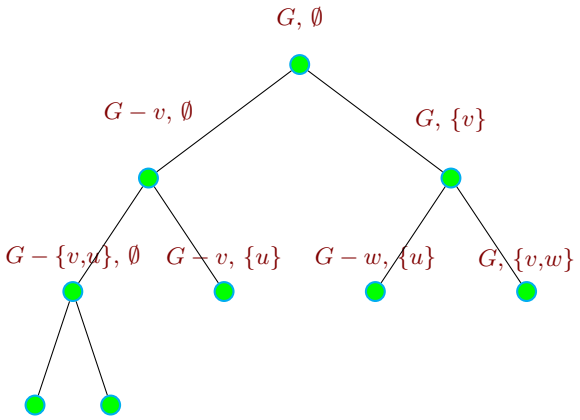
A branching algorithm



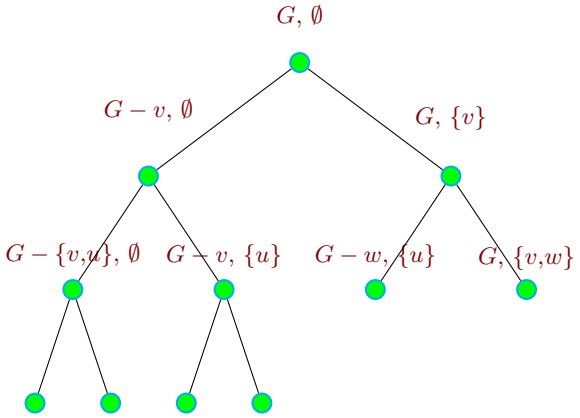
A branching algorithm



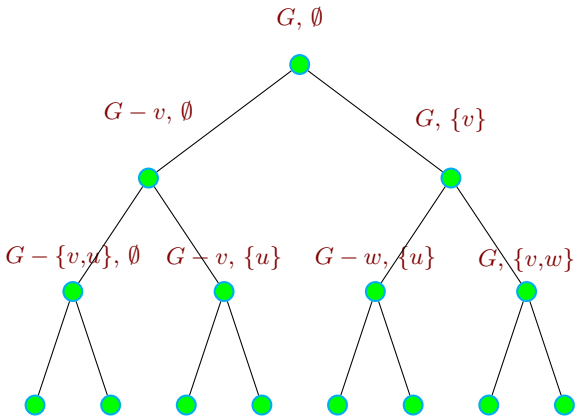
A branching algorithm



A branching algorithm

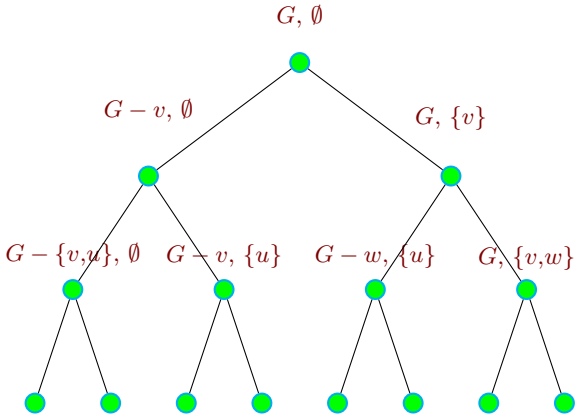


A branching algorithm



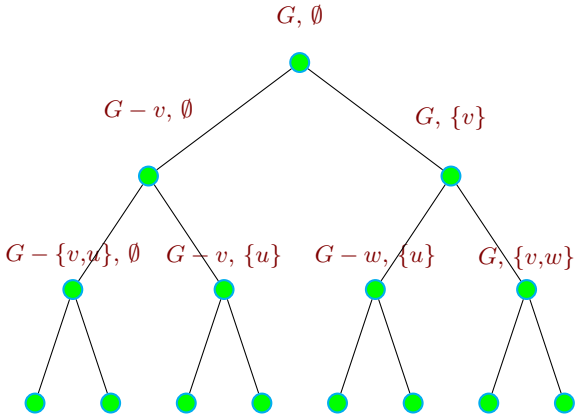
The key of the analysis is to bound the number of nodes, which boils down to bouding the depth.

A branching algorithm



We cannot go left more than k times.
We're hence focused on the right steps.

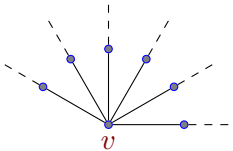
A branching algorithm



Let's fix an execution path.

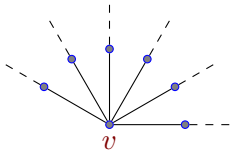
The Analysis

Elementary facts of trees



Elementary facts of trees

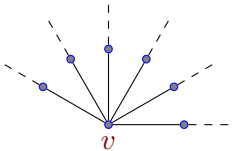
$$\sum_{v \in V(T)} d(v) = 2|E(T)| = 2|V(T)| - 2$$



Elementary facts of trees

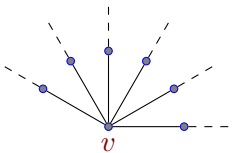
$$\sum_{v \in V(T)} d(v) = 2|E(T)| = 2|V(T)| - 2$$

L: leaves; V_3: vertices of degree ≥ 3.



Elementary facts of trees

$$\sum_{v \in V(T)} d(v) = 2|E(T)| = 2|V(T)| - 2$$

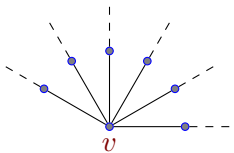


L: leaves; V_3: vertices of degree ≥ 3.

$$-2 = \sum_{v \in L}(d(v)-2) + \sum_{v \in V(T)\setminus L}(d(v)-2) = \sum_{v \in L}(-1) + \sum_{v \in V_3}(d(v)-2) = \sum_{v \in V_3}(d(v)-2) - |L|.$$

Elementary facts of trees

$$\sum_{v \in V(T)} d(v) = 2|E(T)| = 2|V(T)| - 2$$



L: leaves; V_3: vertices of degree ≥ 3.

$$-2 = \sum_{v \in L}(d(v)-2) + \sum_{v \in V(T)\setminus L}(d(v)-2) = \sum_{v \in L}(-1) + \sum_{v \in V_3}(d(v)-2) = \sum_{v \in V_3}(d(v)-2) - |L|.$$

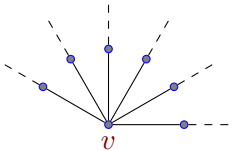
$$\sum_{v \in V_3}(d(v)-2) = |L| - 2.$$

For each $v \notin V_-$,

- either its degree is decreased from ≥ 3 to ≤ 2,
- or leaves are produced to "balance the equation."

Elementary facts of trees

$$\sum_{v \in V(T)} d(v) = 2|E(T)| = 2|V(T)| - 2$$



L: leaves; V_3: vertices of degree ≥ 3.

$$-2 = \sum_{v \in L} (d(v) - 2) + \sum_{v \in V(T) \setminus L} (d(v) - 2) = \sum_{v \in L} (-1) + \sum_{v \in V_3} (d(v) - 2) = \sum_{v \in V_3} (d(v) - 2) - |L|.$$

$$\sum_{v \in V_3} (d(v) - 2) = |L| - 2.$$

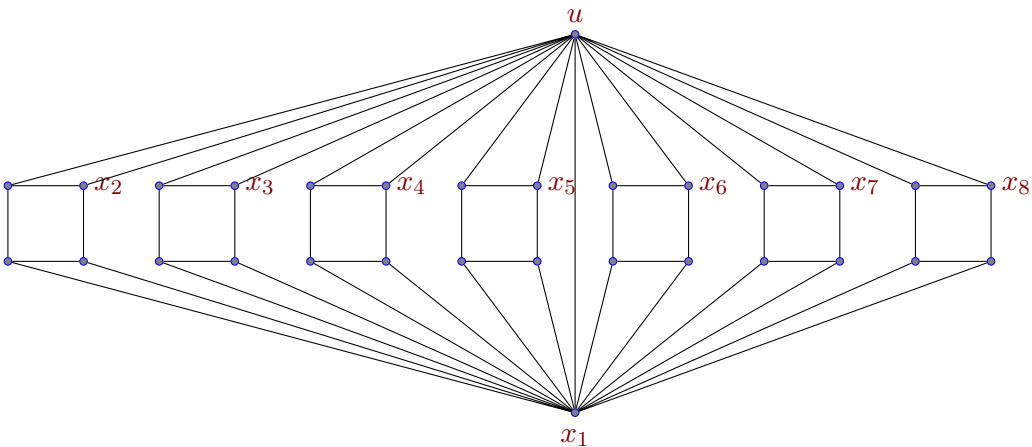
For each $v \notin V_-$,

- either its degree is decreased from ≥ 3 to ≤ 2,
- or leaves are produced to "balance the equation."

This correlates deleted vertices and permanent vertices.

The example

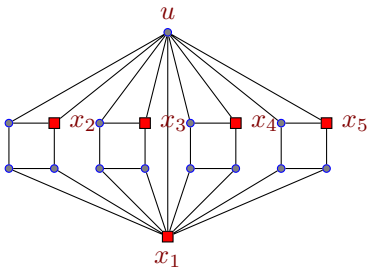
the first vertex x_1 is put into V_-, second u into F,
then the deletion of each x_i decreases its degree by two.



Algorithm invariants

1. During the algorithm, the degree of no vertex can increase.
2. There is no vertex of degree 0 or 1 in the graph when a recursive call is made.

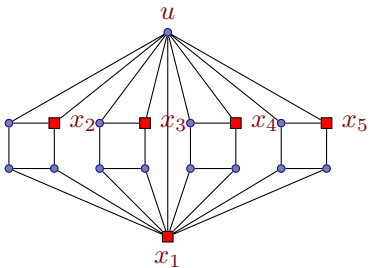
Key definitions and observations



- $d^*(v)$: the degree of v at the moment it is put into V_- or F. $d(v) \geq d^*(v)$
- effective decrements of the degree of a vertex $u \in F$: from $d^*(u)$ to 2.

- an effective decrement is incurred by $x_i \in V_-$ if it is after deleting x_i.
  $\delta(u, x_i)$: #effective decrements of $u \in F$ incurred by $x_i \in V_-$.
  - $\delta(u, x_i)$ may be larger than 1.
  - $\delta(u, x_i) > 0 \quad \Rightarrow \quad u \in F$ when x_i is deleted..

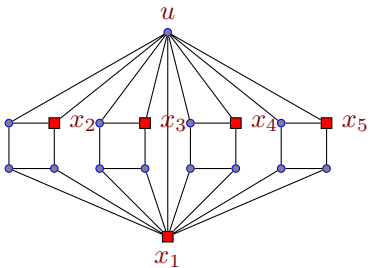
Key definitions and observations



decision points: $x_1 \to u \to x_2 \to x_3 \to x_4 \to x_5$.

- $d^*(v)$: the degree of v at the moment it is put into V_- or F. $\quad d(v) \geq d^*(v)$
- effective decrements of the degree of a vertex $u \in F$: from $d^*(u)$ to 2.
- an effective decrement is incurred by $x_i \in V_-$ if it is after deleting x_i.
  $\delta(u, x_i)$: #effective decrements of $u \in F$ incurred by $x_i \in V_-$.
- $\delta(u, x_i)$ may be larger than 1.
- $\delta(u, x_i) > 0 \quad \Rightarrow \quad u \in F$ when x_i is deleted..

Key definitions and observations

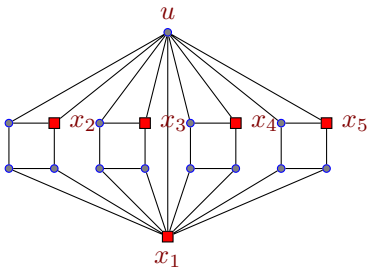


decision points: $x_1 \to u \to x_2 \to x_3 \to x_4 \to x_5$.
$d(u) = 9$; $d^*(u) = 8$

- $d^*(v)$: the degree of v at the moment it is put into V_- or F. $\quad d(v) \geq d^*(v)$
- effective decrements of the degree of a vertex $u \in F$: from $d^*(u)$ to 2.

- an effective decrement is incurred by $x_i \in V_-$ if it is after deleting x_i.
  $\delta(u, x_i)$: #effective decrements of $u \in F$ incurred by $x_i \in V_-$.
- $\delta(u, x_i)$ may be larger than 1.
- $\delta(u, x_i) > 0 \quad \Rightarrow \quad u \in F$ when x_i is deleted. $\quad \Rightarrow \quad d^*(u) \geq d^*(x_i)$.

Key definitions and observations

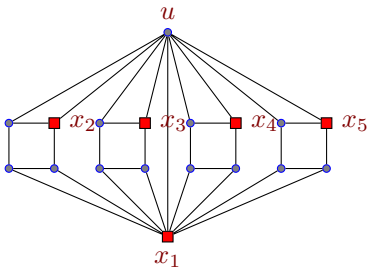


decision points: $x_1 \to u \to x_2 \to x_3 \to x_4 \to x_5$.
$d(u) = 9$; $d^*(u) = 8$
$\delta(u, x_1) = 0$; $\delta(u, x_2) = \delta(u, x_3) = \delta(u, x_4) = 2$; $\delta(u, x_5) = 0$

- $d^*(v)$: the degree of v at the moment it is put into V_- or F. $\quad d(v) \geq d^*(v)$
- effective decrements of the degree of a vertex $u \in F$: from $d^*(u)$ to 2.

- an effective decrement is incurred by $x_i \in V_-$ if it is after deleting x_i.
  $\delta(u, x_i)$: #effective decrements of $u \in F$ incurred by $x_i \in V_-$.
- $\delta(u, x_i)$ may be larger than 1.
- $\delta(u, x_i) > 0 \quad \Rightarrow \quad u \in F$ when x_i is deleted..

Key definitions and observations

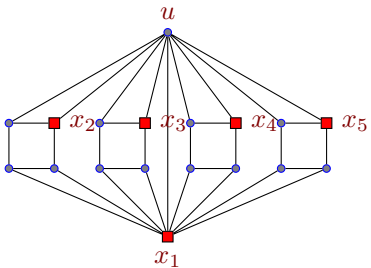


decision points: $x_1 \to u \to x_2 \to x_3 \to x_4 \to x_5$.
$d(u) = 9$; $d^*(u) = 8$
$\delta(u, x_1) = 0$; $\delta(u, x_2) = \delta(u, x_3) = \delta(u, x_4) = 2$; $\delta(u, x_5) = 0$

- $d^*(v)$: the degree of v at the moment it is put into V_- or F. $\quad d(v) \geq d^*(v)$
- effective decrements of the degree of a vertex $u \in F$: from $d^*(u)$ to 2.

- an effective decrement is incurred by $x_i \in V_-$ if it is after deleting x_i.
  $\delta(u, x_i)$: #effective decrements of $u \in F$ incurred by $x_i \in V_-$.
- $\delta(u, x_i)$ may be larger than 1.
- $\delta(u, x_i) > 0 \quad \Rightarrow \quad u \in F$ when x_i is deleted..

Key definitions and observations



decision points: $x_1 \to u \to x_2 \to x_3 \to x_4 \to x_5$.
$d(u) = 9$; $d^*(u) = 8$
$\delta(u, x_1) = 0$; $\delta(u, x_2) = \delta(u, x_3) = \delta(u, x_4) = 2$; $\delta(u, x_5) = 0$

- $d^*(v)$: the degree of v at the moment it is put into V_- or F. $\quad d(v) \geq d^*(v)$
- effective decrements of the degree of a vertex $u \in F$: from $d^*(u)$ to 2.

- an effective decrement is incurred by $x_i \in V_-$ if it is after deleting x_i.
  $\delta(u, x_i)$: #effective decrements of $u \in F$ incurred by $x_i \in V_-$.
- $\delta(u, x_i)$ may be larger than 1.
- $\delta(u, x_i) > 0 \quad \Rightarrow \quad u \in F$ when x_i is deleted..

Lemmas

Lemma 1: For any $u \in F$ and $v \in V_-$, if $\delta(u,v) > 0$ then $d^*(u) \geq d^*(v)$.
Lemma 2: $\sum_{u \in F} \delta(u,v) \leq d^*(v)$ for each $v \in V_-$.

$$|V_-| = \sum_{v \in V_-} 1 = \sum_{v \in V_-} \frac{d^*(v)}{d^*(v)} \geq \sum_{v \in V_-} \frac{1}{d^*(v)} \sum_{u \in F} \delta(u,v)$$

$$= \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(v)} \geq \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(u)} = \sum_{u \in F} \frac{1}{d^*(u)} \sum_{v \in V_-} \delta(u,v)$$

$$= \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{1}{3} = \frac{|F|}{3},$$

Lemma 3: Therefore, an execution path leading to a solution has depth at most $4k$.

Lemmas

Lemma 1: For any $u \in F$ and $v \in V_-$, if $\delta(u,v) > 0$ then $d^*(u) \geq d^*(v)$.
Lemma 2: $\sum_{u \in F} \delta(u,v) \leq d^*(v)$ for each $v \in V_-$.

$$|V_-| = \sum_{v \in V_-} 1 = \sum_{v \in V_-} \frac{d^*(v)}{d^*(v)} \geq \sum_{v \in V_-} \frac{1}{d^*(v)} \sum_{u \in F} \delta(u,v)$$

$$= \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(v)} \geq \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(u)} = \sum_{u \in F} \frac{1}{d^*(u)} \sum_{v \in V_-} \delta(u,v)$$

$$= \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{1}{3} = \frac{|F|}{3},$$

Lemma 3: Therefore, an execution path leading to a solution has depth at most $4k$.

Lemmas

Lemma 1: For any $u \in F$ and $v \in V_-$, if $\delta(u,v) > 0$ then $d^*(u) \geq d^*(v)$.
Lemma 2: $\sum_{u \in F} \delta(u,v) \leq d^*(v)$ for each $v \in V_-$.

$$|V_-| = \sum_{v \in V_-} 1 = \sum_{v \in V_-} \frac{d^*(v)}{d^*(v)} \geq \sum_{v \in V_-} \frac{1}{d^*(v)} \sum_{u \in F} \delta(u,v)$$

$$= \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(v)} \geq \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(u)} = \sum_{u \in F} \frac{1}{d^*(u)} \sum_{v \in V_-} \delta(u,v)$$

$$= \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{1}{3} = \frac{|F|}{3},$$

Lemma 3: Therefore, an execution path leading to a solution has depth at most $4k$.

Lemmas

Lemma 1: For any $u \in F$ and $v \in V_-$, if $\delta(u,v) > 0$ then $d^*(u) \geq d^*(v)$.
Lemma 2: $\sum_{u \in F} \delta(u,v) \leq d^*(v)$ for each $v \in V_-$.

$$|V_-| = \sum_{v \in V_-} 1 = \sum_{v \in V_-} \frac{d^*(v)}{d^*(v)} \geq \sum_{v \in V_-} \frac{1}{d^*(v)} \sum_{u \in F} \delta(u,v)$$

$$= \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(v)} \geq \sum_{v \in V_-} \sum_{u \in F} \frac{\delta(u,v)}{d^*(u)} = \sum_{u \in F} \frac{1}{d^*(u)} \sum_{v \in V_-} \delta(u,v)$$

$$= \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{1}{3} = \frac{|F|}{3},$$

Lemma 3: Therefore, an execution path leading to a solution has depth at most $4k$.

We terminate all execution paths after $4k$ steps $\Rightarrow O(16^k \cdot n^2)$. □

Epilogue

[Furst Gross McGeoch 1988] feedback vertex set on subcubic graphs is in P.
[C. Chen Liu 2010] This can be extended to the setting $d(v) \leq 3$ for $v \in V(G) \setminus F$.

Only put vertices of degree ≥ 4 into F, then

$$|V_-| \geq \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{2}{4} = \frac{|F|}{2}.$$

Epilogue

[Furst Gross McGeoch 1988] feedback vertex set on subcubic graphs is in P.
[C. Chen Liu 2010] This can be extended to the setting $d(v) \leq 3$ for $v \in V(G) \setminus F$.

Only put vertices of degree ≥ 4 into F, then

$$|V_-| \geq \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{2}{4} = \frac{|F|}{2}.$$

Epilogue

[Furst Gross McGeoch 1988] feedback vertex set on subcubic graphs is in P.
[C. Chen Liu 2010] This can be extended to the setting $d(v) \leq 3$ for $v \in V(G) \setminus F$.

Only put vertices of degree ≥ 4 into F, then

$$|V_-| \geq \sum_{u \in F} \frac{d^*(u) - 2}{d^*(u)} \geq \sum_{u \in F} \frac{2}{4} = \frac{|F|}{2}.$$

We terminate all execution paths after $3k$ steps $\Rightarrow O(8^k \cdot n^2)$. $\square$

Final remark

Beauty is the first test:
there is no permanent place in the world for ugly mathematics..

G. H. Hardy

To theorists:
stop ignoring successful heuristic algorithms by pretending their nonexistence!


thanks!