The Decycling Number of Outerplanar Graphs

Hung-Lin Fu

Department of Applied Mathematics,
National Chiao Tung University, Taiwan

This is a joint work with Huilan Chang and Min-Yun Lien.

Outline

- Decycling Problem
- Applications
- Computational Complexity
- Its Variations
- Cycle Packing Problem
- Our Contribution
- Further Work

Definition and Applications

Definition (Decycling Problem)

Given a directed/undirected graph $G = (V, E)$, find a minimum set $D \subset V$ such that $G \setminus D$ is acyclic.

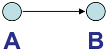
- Has applications in
  - Deadlock prevention in operating systems (Wang et al. 1985; Silberschatz et al. 2003)

Definition and Applications

Definition (Decycling Problem)

Given a directed/undirected graph $G = (V, E)$, find a minimum set $D \subset V$ such that $G \setminus D$ is acyclic.

- Has applications in
  - Deadlock prevention in operating systems (Wang et al. 1985; Silberschatz et al. 2003)
    Example: An operating system schedules different processes.



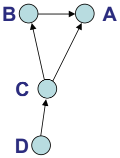
Process A is waiting for the resource on Process B so it can't release its own resource.

Definition and Applications

Definition (Decycling Problem)

Given a directed/undirected graph $G = (V, E)$, find a minimum set $D \subset V$ such that $G \setminus D$ is acyclic.

- Has applications in
  - Deadlock prevention in operating systems (Wang et al. 1985; Silberschatz et al. 2003)
    Example: An operating system schedules different processes.



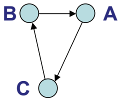
Process A is waiting for no resource so it can release its resource.

Definition and Applications

Definition (Decycling Problem)

Given a directed/undirected graph $G = (V, E)$, find a minimum set $D \subset V$ such that $G \setminus D$ is acyclic.

- Has applications in
  - Deadlock prevention in operating systems (Wang et al. 1985; Silberschatz et al. 2003)
    Example: An operating system schedules different processes.



Deadlock:
Competing actions are each waiting for the other to finish, and thus neither ever does.
Solution:
Remove some processes to **break such cycles** and put them in a waiting queue.

Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)

Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
- ▷ Vertices are colored YES or NO.

Applications

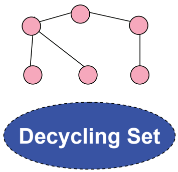
- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - Vertices are colored YES or NO.
  - Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.

Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - Vertices are colored YES or NO.
  - Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.
  - Problem: set the minimum number of vertices YES at beginning such that all vertices become YES eventually.

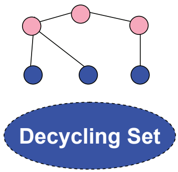
Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - ▷ Vertices are colored YES or NO.
  - ▷ Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.
  - ▷ Problem: set the minimum number of vertices YES at beginning such that all vertices become YES eventually.
  - ▷ A decycling set could be the only choice!
    Example: A 4-regular graph (such as toroidal mesh network).



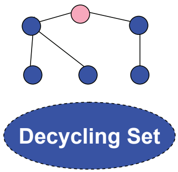
Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - Vertices are colored YES or NO.
  - Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.
  - Problem: set the minimum number of vertices YES at beginning such that all vertices become YES eventually.
  - A decycling set could be the only choice!
    Example: A 4-regular graph (such as toroidal mesh network).



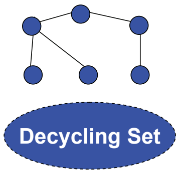
Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - ▷ Vertices are colored YES or NO.
  - ▷ Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.
  - ▷ Problem: set the minimum number of vertices YES at beginning such that all vertices become YES eventually.
  - ▷ A decycling set could be the only choice! Example: A 4-regular graph (such as toroidal mesh network).



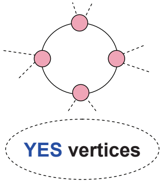
Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - ▷ Vertices are colored YES or NO.
  - ▷ Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.
  - ▷ Problem: set the minimum number of vertices YES at beginning such that all vertices become YES eventually.
  - ▷ A decycling set could be the only choice!
    Example: A 4-regular graph (such as toroidal mesh network).



Applications

- Monopolies in synchronous distributed systems (Peleg 1998; Peleg 2002)
  - ▷ Vertices are colored YES or NO.
  - ▷ Monotone synchronous system: at each step a NO vertex changes to YES if more than half of its neighbors are YES.
  - ▷ Problem: set the minimum number of vertices YES at beginning such that all vertices become YES eventually.
  - ▷ A decycling set could be the only choice!
    Example: A 4-regular graph (such as toroidal mesh network).



NO vertices on the cycle remain NO.
Failed!

Applications

- Constraint satisfaction problem (Dechter 1990).
- Bayesian inference in artificial intelligence (Bar-Yehuda et al. 1998).
- Converters' placement problem in optical networks (Kleinberg and Kumar 1999).
- VLSI chip design (Festa et al. 2000).

Complexity

- Has been extensively studied.
  - NP-hard
    Reduction from VERTEX COVER (R. Karp 1972).
    Even for planar graphs.

Complexity

- Has been extensively studied.
  - NP-hard
    Reduction from VERTEX COVER (R. Karp 1972).
    Even for planar graphs.
  - APX-complete
    2-approximation algorithm (Becker and Geiger 1996)
    Reduction from VERTEX COVER.

Complexity

- Has been extensively studied.
  - NP-hard
    Reduction from VERTEX COVER (R. Karp 1972).
    Even for planar graphs.
  - APX-complete
    2-approximation algorithm (Becker and Geiger 1996)
    Reduction from VERTEX COVER.
  - Exact algorithm
    $O(1.7347^n n^{O(1)})$ time on n-vertex undirected graphs (Fomin et al. 2008)

Complexity

- Parameterized version

▷ Given a graph G and a parameter k, either construct a decycling set of size bounded by k or report no such decycling set.

Complexity

- Parameterized version
- Given a graph G and a parameter k, either construct a decycling set of size bounded by k or report no such decycling set.
- Parameterized algorithms ($O(f(k)n^{O(1)})$ time):
- Exact algorithm: $O(3.83^k kn^2)$ time (Cao et al. 2010).

Complexity

- Parameterized version
  - Given a graph G and a parameter k, either construct a decycling set of size bounded by k or report no such decycling set.
  - Parameterized algorithms ($O(f(k)n^{O(1)})$ time):
    - Exact algorithm: $O(3.83^k kn^2)$ time (Cao et al. 2010).
    - Quadratic kernel $|E| = O(k^2)$ (Thomasse 2009; Dell and van Melkebeek 2010).

Complexity

- Parameterized version
- ▷ Given a graph G and a parameter k, either construct a decycling set of size bounded by k or report no such decycling set.
- ▷ Parameterized algorithms ($O(f(k)n^{O(1)})$ time):
  - Exact algorithm: $O(3.83^k kn^2)$ time (Cao et al. 2010).
  - Quadratic kernel $|E| = O(k^2)$ (Thomasse 2009; Dell and van Melkebeek 2010).
  - Linear kernel $|E| = O(k)$ for planar graph (Bodlaender 2008).

Related Problems

- Is related or equivalent to
  - Feedback vertex set problem (Wang et al. 1985).
  - Hitting cycle problem

Related Problems

- Is related or equivalent to
  - Feedback vertex set problem (Wang et al. 1985).
  - Hitting cycle problem
  - Maximum induced forest problem
    (Erdös, Saks and Sós 1986 Maximum induced trees in graphs).
    At least $\frac{71n+72}{128}$ for triangle-tree n-vertex planar graphs (Kowalik et al. 2010).

Variations

- Has the following variations (survey Festa et al. 2000).
  - Graph Bipartization Problem
    Find $D \subset E$ such that $G \setminus D$ has no odd cycle.

Variations

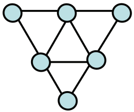
- Has the following variations (survey Festa et al. 2000).
  - Graph Bipartization Problem
    Find $D \subset E$ such that $G \setminus D$ has no odd cycle.
  - Weighted Decycling Problem
    Seek a minimum-weight decycling set D where each vertex has a weight.

Variations

- Has the following variations (survey Festa et al. 2000).
  - Graph Bipartization Problem
    Find $D \subset E$ such that $G \setminus D$ has no odd cycle.
  - Weighted Decycling Problem
    Seek a minimum-weight decycling set D where each vertex has a weight.
  - Loop Cut Set Problem
    Given $A(C) \subseteq V(C)$ for each cycle C of G, find a minimum set D such that $D \cap A(C) \neq \emptyset$.

Relations with Cycle Packing Number

- Is often compared with the following graph parameter.
  - Cycle Packing Number $c(G)$: the maximum number of vertex-disjoint cycles of G.



$c(G)=1$

  - **Definition:** $\tau(G) :=$ the decycling number (minimum size of decycling set) of G.

Relations with Cycle Packing Number

- Dirac and Gallai had interest in the relations between $c(G)$ and $\tau(G)$.
- It is clear that $c(G) \leq \tau(G)$.

Relations with Cycle Packing Number

- Dirac and Gallai had interest in the relations between $c(G)$ and $\tau(G)$.
- It is clear that $c(G) \leq \tau(G)$.

- **Definition:** $\tau(k) := \max\{\tau(G) : c(G) = k\}$.

Relations with Cycle Packing Number

- Dirac and Gallai had interest in the relations between $c(G)$ and $\tau(G)$.
- It is clear that $c(G) \leq \tau(G)$.

- **Definition:** $\tau(k) := \max\{\tau(G) : c(G) = k\}$.
- $\tau(1) = 3$; $c(K_5) = 1$ and $\tau(K_5) = 3$ (Bollobás 1964).
- $\tau(2) = 6$ and $9 \leq \tau(3) \leq 12$ (Voss 1968).

Relations with Cycle Packing Number

- Dirac and Gallai had interest in the relations between $c(G)$ and $\tau(G)$.
- It is clear that $c(G) \leq \tau(G)$.

- **Definition:** $\tau(k) := \max\{\tau(G) : c(G) = k\}$.
- $\tau(1) = 3$; $c(K_5) = 1$ and $\tau(K_5) = 3$ (Bollob*á*s 1964).
- $\tau(2) = 6$ and $9 \leq \tau(3) \leq 12$ (Voss 1968).
- $c_1 k \log k \leq \tau(k) \leq c_2 k \log k$ for some constants c_1 and c_2 (Erd*ö*s and P*ó*sa 1964).

Relations with Cycle Packing Number

- Consider planar graphs:

Relations with Cycle Packing Number

- Consider planar graphs:

Jones' Conjecture (Kloks, Lee and Liu 2002)

For every planar graph G, $\tau(G) \leq 2c(G)$.

Relations with Cycle Packing Number

- Consider planar graphs:

Jones' Conjecture (Kloks, Lee and Liu 2002)

For every planar graph G, $\tau(G) \leq 2c(G)$.

Theorem (Chen, Fu and Shih 2010)

For every planar graph G, $\tau(G) \leq 3c(G)$.

Our Contribution

Consider outerplanar graphs:

Theorem (Kloks, Lee and Liu 2002)

For every outerplanar graph G, $\tau(G) \leq 2c(G)$.

- An outerplanar graph G is called *lower-extremal* if $\tau(G) = c(G)$ and *upper-extremal* if $\tau(G) = 2c(G)$.

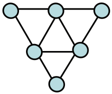
Our Contribution

Consider outerplanar graphs:

Theorem (Kloks, Lee and Liu 2002)

For every outerplanar graph G, $\tau(G) \leq 2c(G)$.

- An outerplanar graph G is called *lower-extremal* if $\tau(G) = c(G)$ and *upper-extremal* if $\tau(G) = 2c(G)$.
- Upper-Extremal Results:
  - We define a *sun graph* S_3 as follows.



  - $\tau(S_3) = 2 = 2c(S_3)$.

Our Contribution

Theorem (Chang, Fu, Lien, 2011)

An outerplanar graph G is upper-extremal if and only if G is an S_3-tree.

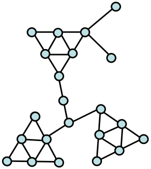
- A graph is an *S_3-tree of order t* if it has exactly t vertex-disjoint S_3-subdivisions and every edge not on these S_3-subdivisions belongs to no cycle.

Our Contribution

Theorem (Chang, Fu, Lien, 2011)

An outerplanar graph G is upper-extremal if and only if G is an S_3-tree.

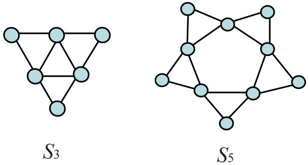
- A graph is an *S_3-tree of order t* if it has exactly t vertex-disjoint S_3-subdivisions and every edge not on these S_3-subdivisions belongs to no cycle.
- Example:



An S_3-tree G of order 3, where $\tau(G) = 6 = 2c(G)$.

Our Contribution

- Lower-Extremal Results:
  - The following graphs are NOT lower-extremal ($\tau(G) \neq c(G)$):
    Sun graphs S_k with odd number k:



  - $\tau(S_k) = \lceil \frac{k}{2} \rceil$ and $c(S_k) = \lfloor \frac{k}{2} \rfloor$.

Our Contribution

Theorem (Chang, Fu, Lien, 2011)

For an outerplanar graph G, if G has no S_k-subdivision for all odd number k, then G is lower-extremal.

Our Contribution

Theorem (Chang, Fu, Lien, 2011)

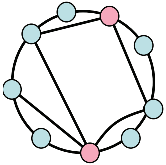
For an outerplanar graph G, if G has no S_k-subdivision for all odd number k, then G is lower-extremal.

Lemma

If G is a 2-connected outerplanar graph with no S_k-subdivision for all odd number k, then G is lower-extremal.

(Proved by induction on $|E(G)|$.)

- Example:



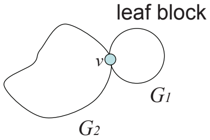
Our Contribution

- A graph property is called *monotone* if it is closed under removal of vertices.
  - ▷ ”Without S_k-subdivision for all odd number k” is a monotone graph property.

Lemma

Suppose that a 2-connected graph is lower-extremal provided that it satisfies a monotone property $\mathcal{P}$. Then a graph is lower-extremal if it satisfies $\mathcal{P}$.

Proof. By induction on $|G|$. Suppose G satisfies $\mathcal{P}$. Consider that G has a cut vertex v.



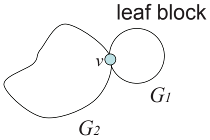
Our Contribution

- A graph property is called *monotone* if it is closed under removal of vertices.
  - ▷ "Without S_k-subdivision for all odd number k" is a monotone graph property.

Lemma

Suppose that a 2-connected graph is lower-extremal provided that it satisfies a monotone property $\mathcal{P}$. Then a graph is lower-extremal if it satisfies $\mathcal{P}$.

Proof. By induction on $|G|$. Suppose G satisfies $\mathcal{P}$. Consider that G has a cut vertex v.



$c(G) = c(G_1) + c(G_2)$ or
$c(G_1) + c(G_2) - 1$.
$\tau(G) \leq \tau(G_1) + \tau(G_2)$.

Our Contribution

Suppose to the contrary that $\tau(G) > c(G)$. Then
(1) $c(G) = c(G_1) + c(G_2) - 1$ and
(2) $\tau(G) = \tau(G_1) + \tau(G_2)$.

Our Contribution

Suppose to the contrary that $\tau(G) > c(G)$. Then
(1) $c(G) = c(G_1) + c(G_2) - 1$ and
(2) $\tau(G) = \tau(G_1) + \tau(G_2)$.
(1) $\Rightarrow$ every maximum set of vertex-disjoint cycles of G_i must contain a cycle with v for $i = 1, 2$.
Thus $c(G_i - v) < c(G_i)$ for $i = 1, 2$.

Our Contribution

Suppose to the contrary that $\tau(G) > c(G)$. Then
(1) $c(G) = c(G_1) + c(G_2) - 1$ and
(2) $\tau(G) = \tau(G_1) + \tau(G_2)$.
(1) $\Rightarrow$ every maximum set of vertex-disjoint cycles of G_i must contain a cycle with v for $i = 1, 2$.
Thus $c(G_i - v) < c(G_i)$ for $i = 1, 2$.
(2) $\Rightarrow$ v does not belong to any minimum decycling set of G^* where $G^* = G_1$ or G_2.
Thus $\tau(G^* - v) = \tau(G^*)$.

Our Contribution

Suppose to the contrary that $\tau(G) > c(G)$. Then
(1) $c(G) = c(G_1) + c(G_2) - 1$ and
(2) $\tau(G) = \tau(G_1) + \tau(G_2)$.
(1) $\Rightarrow$ every maximum set of vertex-disjoint cycles of G_i must contain a cycle with v for $i = 1, 2$.
Thus $c(G_i - v) < c(G_i)$ for $i = 1, 2$.
(2) $\Rightarrow$ v does not belong to any minimum decycling set of G^* where $G^* = G_1$ or G_2.
Thus $\tau(G^* - v) = \tau(G^*)$.
By the monotonicity of $\mathcal{P}$ and the induction hypothesis,
$c(G^* - v) = \tau(G^* - v) = \tau(G^*) = c(G^*)$, a contradiction. ∎

Further Work

- For a planar graph G, $\tau(G) \overset{?}{\leq} 2c(G)$ (Jones' conjecture 2002).
- For a planar graph G, $\tau(G) \overset{?}{\leq} |V(G)|/2$ (Albertson and Berman 1979).
- For a bipartite planar graph G, $\tau(G) \overset{?}{\leq} 3|V(G)|/8$ (Albertson and Berman 1979).

References

- Albertson MO, Berman DM (1979) A conjecture on planar graphs, Bondy JA, Murty USR, Graph theory and related topics 357.
- Chang H, Fu HL, Lien MY (2011) The decycling number of outerplanar graphs. Submitted.
- Festa P, Pardalos PM, Resende MGC (2000) Feedback set problems, Handbook of Combinatorial Optimization, Du D-Z, Pardalos PM, Eds, Kluwer Academic Publishers, Supplement A, pp 209-259.
- Kloks T, Lee C-M, Liu J (2002) New algorithms for k-face cover, k-feedback vertex set, and k-disjoint cycles on plane and planar graphs. in Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2002) Springer-Verlag, 2573:282-295. (Jones' Conjecture)

Thank you for your attention!