Further Results on the MSVC Problem

Ralucca Gera * Steven Horton ** Craig Rasmussen *
Pantelimon Stănică *

*Naval Postgraduate School, Monterey, CA

**United States Military Academy, West Point, NY

United States Military Academy
18 November 2006

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Definitions

- The graph $G = (V, E)$ is nonempty, undirected, with $|V| = n$.
- The mapping $f : V \to \{1, 2, \ldots, n\}$ is an ordering of V.
- The weight function $w : E \to \mathbb{Z}^+$ is $w(uv) = \min\{f(u), f(v)\}$.
- The min-sum vertex cover number (msvc number) of G is $\mu_s(G) = \min \sum_{e \in E} w(e)$, where the minimum is taken over all $n!$ orderings $f(V)$. We refer to $\mu_s(G)$ as the cost of G.
- A cost set S_f for a given $f(V)$ is defined by

$$S_f = \{u \in V | \exists e \in E \text{ such that } w(e) = f(u)\}.$$

Definitions

- The graph $G = (V, E)$ is nonempty, undirected, with $|V| = n$.
- The mapping $f : V \to \{1, 2, \ldots, n\}$ is an ordering of V.
- The weight function $w : E \to \mathbb{Z}^+$ is $w(uv) = \min\{f(u), f(v)\}$.
- The min-sum vertex cover number (msvc number) of G is $\mu_s(G) = \min \sum_{e \in E} w(e)$, where the minimum is taken over all $n!$ orderings $f(V)$. We refer to $\mu_s(G)$ as the cost of G.
- A cost set S_f for a given $f(V)$ is defined by

$$S_f = \{u \in V | \exists e \in E \text{ such that } w(e) = f(u)\}.$$

Definitions

- The graph $G = (V, E)$ is nonempty, undirected, with $|V| = n$.
- The mapping $f : V \to \{1, 2, \ldots, n\}$ is an ordering of V.
- The weight function $w : E \to \mathbb{Z}^+$ is $w(uv) = \min\{f(u), f(v)\}$.
- The min-sum vertex cover number (msvc number) of G is $\mu_s(G) = \min \sum_{e \in E} w(e)$, where the minimum is taken over all $n!$ orderings $f(V)$. We refer to $\mu_s(G)$ as the cost of G.
- A cost set S_f for a given $f(V)$ is defined by

$$S_f = \{u \in V | \exists e \in E \text{ such that } w(e) = f(u)\}.$$

Definitions

- The graph $G = (V, E)$ is nonempty, undirected, with $|V| = n$.
- The mapping $f : V \to \{1, 2, \ldots, n\}$ is an ordering of V.
- The weight function $w : E \to \mathbb{Z}^+$ is $w(uv) = \min\{f(u), f(v)\}$.
- The min-sum vertex cover number (msvc number) of G is $\mu_s(G) = \min \sum_{e \in E} w(e)$, where the minimum is taken over all $n!$ orderings $f(V)$. We refer to $\mu_s(G)$ as the cost of G.
- A cost set S_f for a given $f(V)$ is defined by

$$S_f = \{u \in V | \exists e \in E \text{ such that } w(e) = f(u)\}.$$

Definitions

- The graph $G = (V, E)$ is nonempty, undirected, with $|V| = n$.
- The mapping $f : V \to \{1, 2, \ldots, n\}$ is an ordering of V.
- The weight function $w : E \to \mathbb{Z}^+$ is $w(uv) = \min\{f(u), f(v)\}$.
- The min-sum vertex cover number (msvc number) of G is $\mu_s(G) = \min \sum_{e \in E} w(e)$, where the minimum is taken over all $n!$ orderings $f(V)$. We refer to $\mu_s(G)$ as the cost of G.
- A cost set S_f for a given $f(V)$ is defined by

$$S_f = \{u \in V | \exists e \in E \text{ such that } w(e) = f(u)\}.$$

Motivation: Data Migration

- The msvc problem is motivated by a data migration problem. (e.g., Feige, Lovász, Tetali (2004); Kim (2004))
- The graph version: vertices represent servers, with label on vertex s dictating time at which server s moves data.
- Edges represent file transfers. Unweighted problem (unit edge weights) assumes constant-time transfers.
- Goal: minimize average service delay over all transfers.
- NP-Complete, even with uniform edge weights.
- Our initial focus is on msvc numbers of garden-variety graph classes.

Motivation: Data Migration

- The msvc problem is motivated by a data migration problem. (e.g., Feige, Lovász, Tetali (2004); Kim (2004))
- The graph version: vertices represent servers, with label on vertex s dictating time at which server s moves data.
- Edges represent file transfers. Unweighted problem (unit edge weights) assumes constant-time transfers.
- Goal: minimize average service delay over all transfers.
- NP-Complete, even with uniform edge weights.
- Our initial focus is on msvc numbers of garden-variety graph classes.

Motivation: Data Migration

- The msvc problem is motivated by a data migration problem. (e.g., Feige, Lovász, Tetali (2004); Kim (2004))
- The graph version: vertices represent servers, with label on vertex s dictating time at which server s moves data.
- Edges represent file transfers. Unweighted problem (unit edge weights) assumes constant-time transfers.
- Goal: minimize average service delay over all transfers.
- NP-Complete, even with uniform edge weights.
- Our initial focus is on msvc numbers of garden-variety graph classes.

Motivation: Data Migration

- The msvc problem is motivated by a data migration problem. (e.g., Feige, Lovász, Tetali (2004); Kim (2004))
- The graph version: vertices represent servers, with label on vertex s dictating time at which server s moves data.
- Edges represent file transfers. Unweighted problem (unit edge weights) assumes constant-time transfers.
- Goal: minimize average service delay over all transfers.
- NP-Complete, even with uniform edge weights.
- Our initial focus is on msvc numbers of garden-variety graph classes.

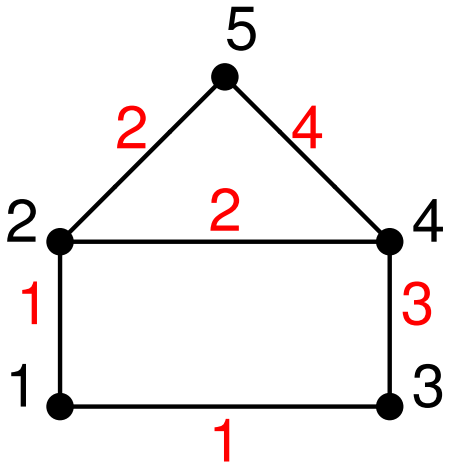
Motivation: Data Migration

- The msvc problem is motivated by a data migration problem. (e.g., Feige, Lovász, Tetali (2004); Kim (2004))
- The graph version: vertices represent servers, with label on vertex s dictating time at which server s moves data.
- Edges represent file transfers. Unweighted problem (unit edge weights) assumes constant-time transfers.
- Goal: minimize average service delay over all transfers.
- NP-Complete, even with uniform edge weights.
- Our initial focus is on msvc numbers of garden-variety graph classes.

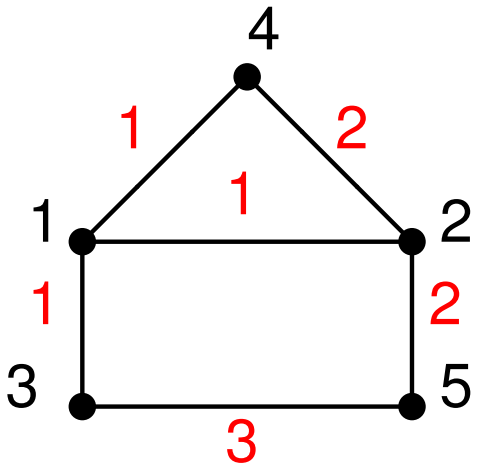
Motivation: Data Migration

- The msvc problem is motivated by a data migration problem. (e.g., Feige, Lovász, Tetali (2004); Kim (2004))
- The graph version: vertices represent servers, with label on vertex s dictating time at which server s moves data.
- Edges represent file transfers. Unweighted problem (unit edge weights) assumes constant-time transfers.
- Goal: minimize average service delay over all transfers.
- NP-Complete, even with uniform edge weights.
- Our initial focus is on msvc numbers of garden-variety graph classes.

Concrete Examples



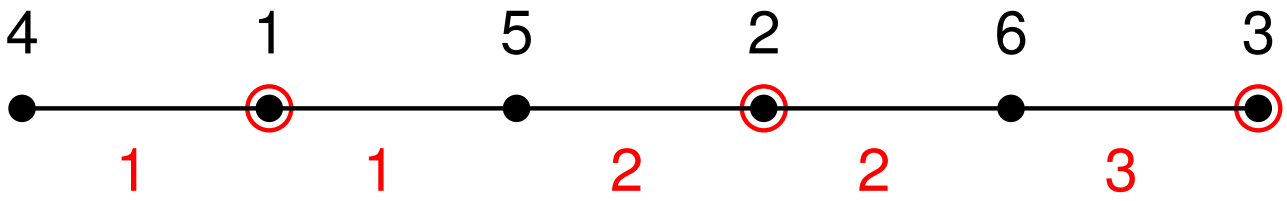
$\text{Cost}(f) = 13$



$\text{Cost}(f) = 10$

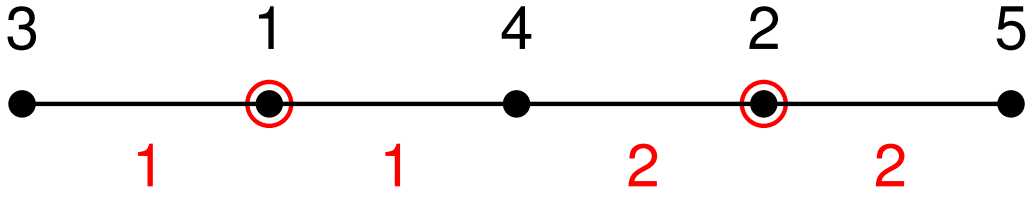
Concrete Examples

P_6:



$$\mu_s(P_6) = 9$$

P_5:



$$\mu_s(P_5) = 6$$

Immediate Bounds

- A cost set must be a vertex cover, so $\beta(G) \leq \mu_s(G)$. (This is sharp only for K_2.)
- A connected graph G on n vertices has at least $n-1$ edges, so $n-1 \leq \mu_s(G)$. (Sharp for $K_{1,n-1}$.)
- Given a connected graph G on n vertices, the vertex labeled i in any optimal cost set contributes $w(e) = f(i)$ to at most $n-i$ edges ($1 \leq i < n$), so

$$\mu_s(G) \leq \sum_{i=1}^{n-1} i(n-i) = \frac{n^3 - n}{6}. \text{ (Sharp for } K_n\text{)}$$

Immediate Bounds

- A cost set must be a vertex cover, so $\beta(G) \leq \mu_s(G)$. (This is sharp only for K_2.)
- A connected graph G on n vertices has at least $n-1$ edges, so $n-1 \leq \mu_s(G)$. (Sharp for $K_{1,n-1}$.)
- Given a connected graph G on n vertices, the vertex labeled i in any optimal cost set contributes $w(e) = f(i)$ to at most $n-i$ edges ($1 \leq i < n$), so

$$\mu_s(G) \leq \sum_{i=1}^{n-1} i(n-i) = \frac{n^3-n}{6}. \text{ (Sharp for } K_n\text{)}$$

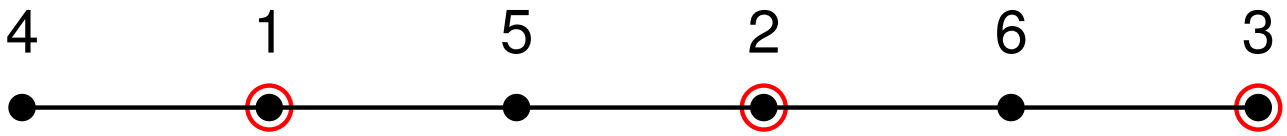
Immediate Bounds

- A cost set must be a vertex cover, so $\beta(G) \leq \mu_s(G)$. (This is sharp only for K_2.)
- A connected graph G on n vertices has at least $n-1$ edges, so $n-1 \leq \mu_s(G)$. (Sharp for $K_{1,n-1}$.)
- Given a connected graph G on n vertices, the vertex labeled i in any optimal cost set contributes $w(e) = f(i)$ to at most $n-i$ edges ($1 \leq i < n$), so

$$\mu_s(G) \leq \sum_{i=1}^{n-1} i(n-i) = \frac{n^3 - n}{6}. \text{ (Sharp for } K_n\text{)}$$

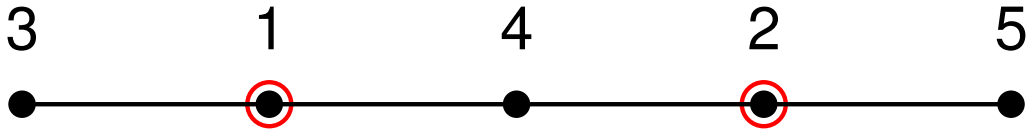
- $\mu_s(P_n) = \left\lfloor \frac{n^2}{4} \right\rfloor$. In the case $n = 2k + 1$, the independent set of k internal vertices is an msvc cost set; in the case $n = 2k$, we use $k - 1$ internal vertices and one pendant.

P_6:



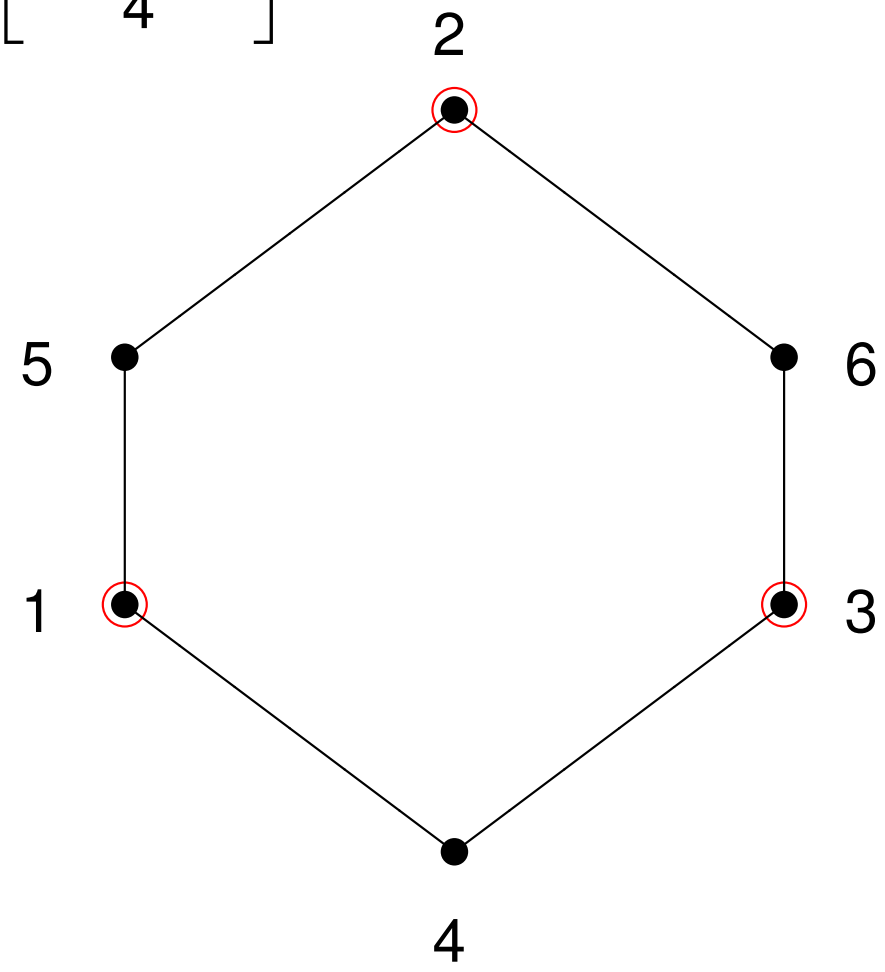
$\mu_s(P_6) = 9$

P_5:



$\mu_s(P_5) = 6$

- $\mu_s(C_n) = \left\lfloor \frac{(n+1)^2}{4} \right\rfloor$.



$\mu_s(C_6) = 12$

Some Exact Results

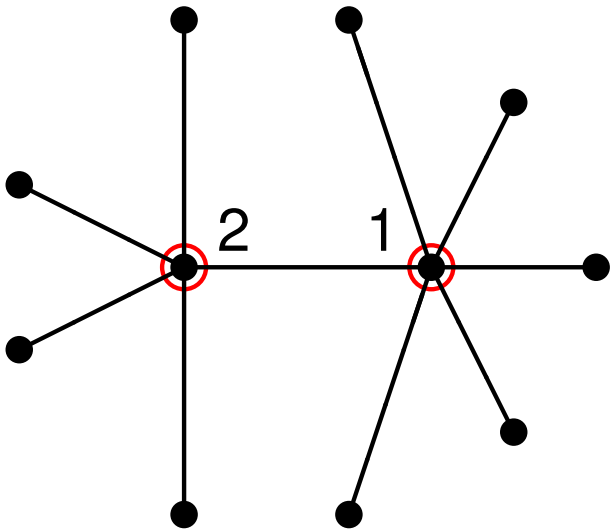
- If $a \leq b$, then $\mu_s(K_{a,b}) = b\sum_{i=1}^{a} i = \frac{ba(a+1)}{2}$.
- If $a \leq b$, then the double star $(S(a,b))$ has $\mu_s(S(a,b)) = 2a + b - 2$.

$\mu_s(S(5,6)) = 14$

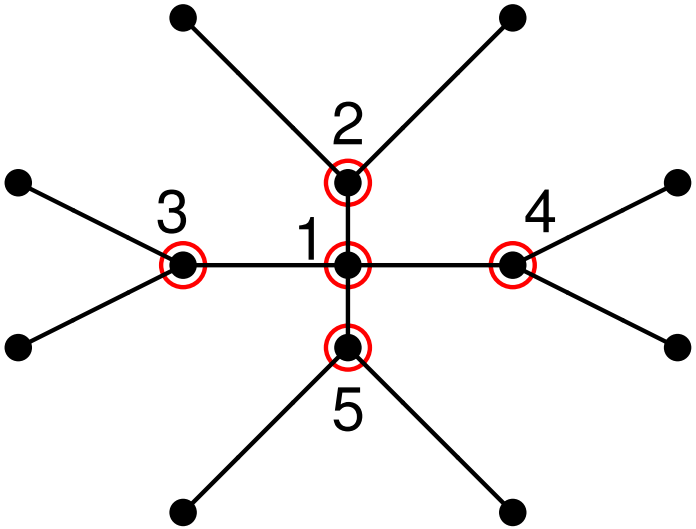
Some Exact Results

- If $a \leq b$, then $\mu_s(K_{a,b}) = b\sum_{i=1}^{a} i = \frac{ba(a+1)}{2}$.
- If $a \leq b$, then the double star $(S(a,b))$ has $\mu_s(S(a,b)) = 2a + b - 2$.

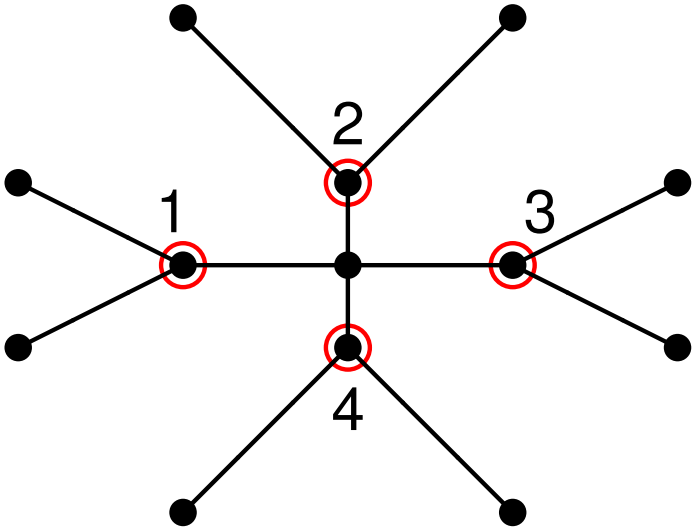
$\mu_s(S(5,6)) = 14$



Dark Days for the Greedy Approach



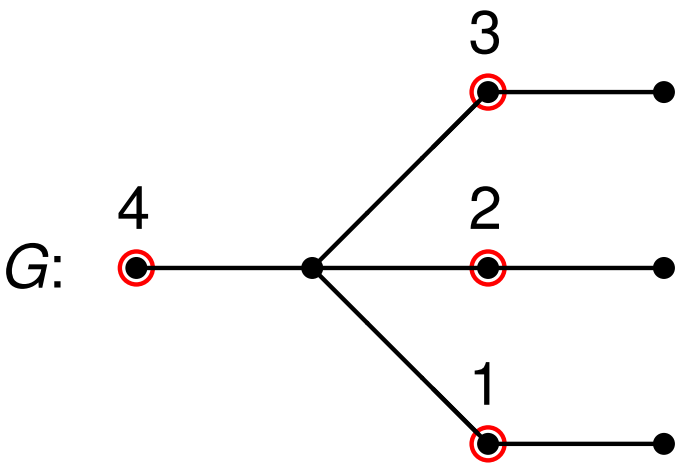
$$\sum_{e \in E} w(e) = 32$$

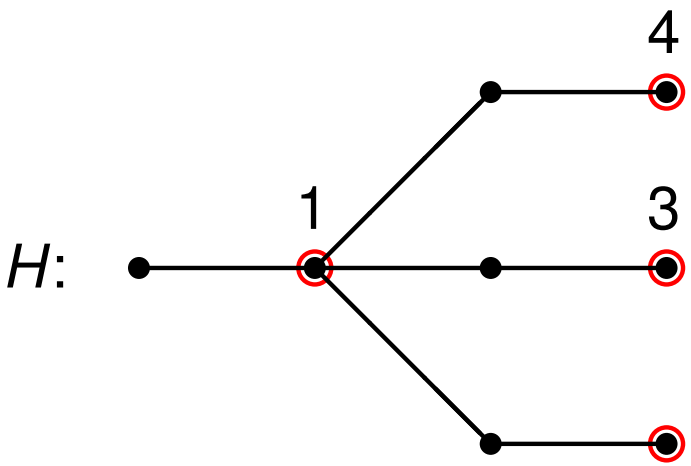


$$\sum_{e \in E} w(e) = 30$$

Not Just Any Independent Set Will Do

- If G has an independent vertex cover, an upper bound on $\mu_s(G)$ can be computed, but not every independent vertex cover constitutes a minimum cost set.





Cost $\sum_{e \in E} w(e) = 16$

Cost $\sum_{e \in E} w(e) = 13 = \mu_s(G)$

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

A General Result for Regular Graphs

- Suppose G is k-regular, with $k \geq 2$. Then

$$k\sum_{i=1}^{\alpha} i = \frac{k\alpha(\alpha+1)}{2} \leq \mu_s \leq \frac{k\beta(\beta+1)}{2} = k\sum_{i=1}^{\beta} i,$$

  where $\alpha = \alpha(G)$ is the independence number of G and $\beta = \beta(G)$ is the vertex covering number of G.

- This gives an exact result when $\alpha(G) = \beta(G)$, in which case

$$\mu_s(G) = \frac{kn(n+1)}{2}.$$

A General Result for Regular Graphs

- Suppose G is k-regular, with $k \geq 2$. Then

$$k\sum_{i=1}^{\alpha} i = \frac{k\alpha(\alpha+1)}{2} \leq \mu_s \leq \frac{k\beta(\beta+1)}{2} = k\sum_{i=1}^{\beta} i,$$

  where $\alpha = \alpha(G)$ is the independence number of G and $\beta = \beta(G)$ is the vertex covering number of G.
- This gives an exact result when $\alpha(G) = \beta(G)$, in which case

$$\mu_s(G) = \frac{kn(n+1)}{2}.$$

Examples: Regular Graphs

- $\mu_s(K_{n,n}) = \dfrac{n^3 + n^2}{2}$.
- For $n \geq 2$, $\mu_s(Q_n) = n2^{n-2}\left(2^{n-1} - 1\right)$.
- For $n \geq 2$, the prism $G \cong C_n \times K_2$ has $\mu_s(G) = \dfrac{3n(n+1)}{2}$ if n is even, and $\mu_s(G) = \dfrac{3n^2 + 3n + 2}{2}$ if n is odd.

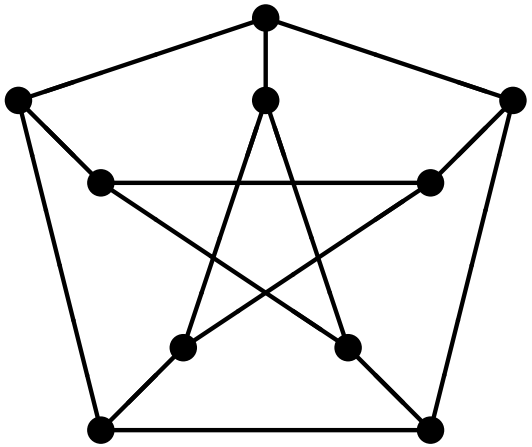
Examples: Regular Graphs

- $\mu_s(K_{n,n}) = \dfrac{n^3 + n^2}{2}$.
- For $n \geq 2$, $\mu_s(Q_n) = n2^{n-2}\left(2^{n-1} - 1\right)$.
- For $n \geq 2$, the prism $G \cong C_n \times K_2$ has $\mu_s(G) = \dfrac{3n(n+1)}{2}$ if n is even, and $\mu_s(G) = \dfrac{3n^2 + 3n + 2}{2}$ if n is odd.

Examples: Regular Graphs

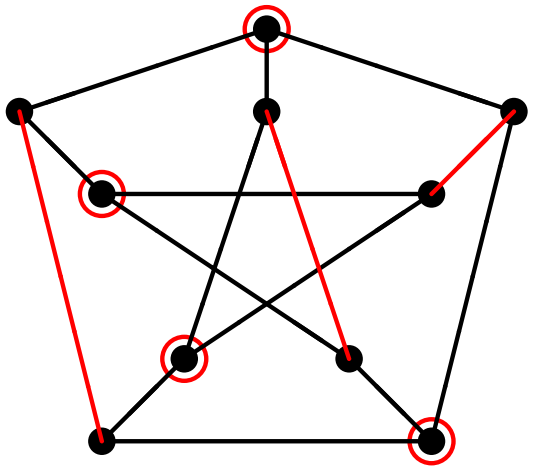
- $\mu_s(K_{n,n}) = \dfrac{n^3 + n^2}{2}$.
- For $n \geq 2$, $\mu_s(Q_n) = n2^{n-2}\left(2^{n-1} - 1\right)$.
- For $n \geq 2$, the prism $G \cong C_n \times K_2$ has $\mu_s(G) = \dfrac{3n(n+1)}{2}$ if n is even, and $\mu_s(G) = \dfrac{3n^2 + 3n + 2}{2}$ if n is odd.

The Petersen Graph



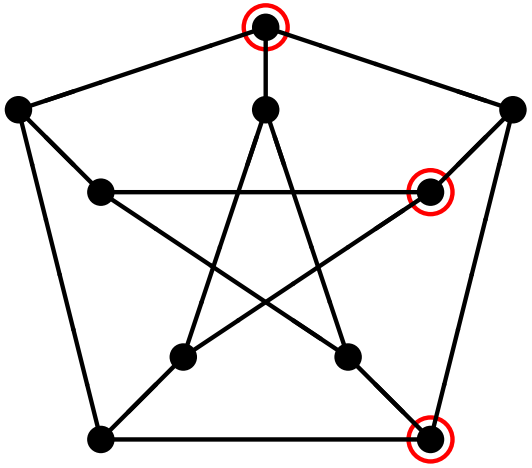
- The proof that the Petersen graph has msvc number 48 provides a nice illustration of determining the msvc number of a graph for which we have no available formula. We consider the numbers of vertices in a cost set that cover three edges.

The Petersen Graph



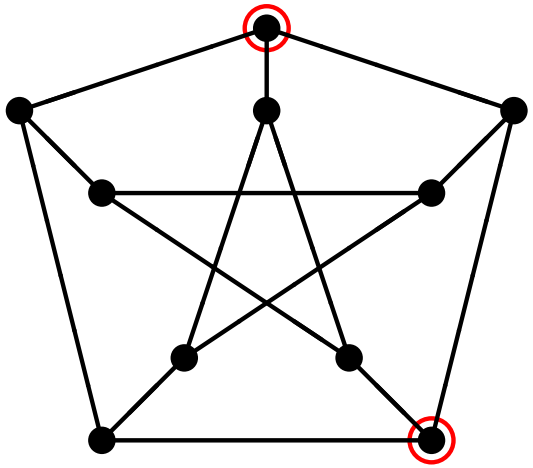
- If S_f contains four independent vertices with available degree three, we have a matching of size three left over, so $\text{Cost}(f) = 3 + 6 + 9 + 12 + 5 + 6 + 7 = 48$.

The Petersen Graph



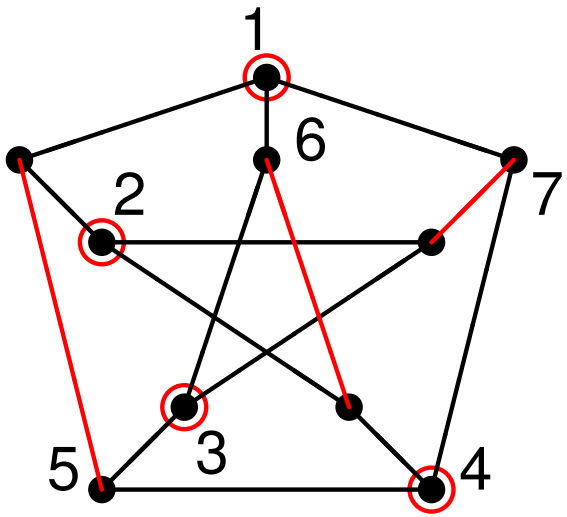
- If we use three independent vertices with available degree three, we have a six edges left over, so $\text{Cost}(f) \geq 3 + 6 + 9 + 8 + 10 + 12 \geq 48$; this is sharp.

The Petersen Graph



- If we use fewer independent vertices with available degree three, the cost rises. For example, if S_f contains only two vertices of degree three, there are nine more edges, and $\text{Cost}(f) \geq 3 + 6 + 6 + 8 + 10 + 12 + 7 = 52$. So no matter what mapping f we choose, $\text{Cost}(f) \geq 48$.

$\mu_s(PG) = 48$



- $\text{Cost}(f) = \mu_s(PG) = 48.$

Biregular Graphs

- A graph $G = (V, E)$ is biregular if $V = V_1 + V_2$, where V_1 is regular of degree j and V_2 is regular of degree k for some j, k. We call G strictly biregular if $j \neq k$.
- A graph $G = (X, Y, E)$ is bipartite biregular if G is both bipartite and biregular.
- If $G = (X, Y, E)$ is bipartite biregular, and if the degree partition coincides with the partition $V = X + Y$, then $j|X| = k|Y|$, and it follows that if $j \geq k$, then

$$\mu_s(G) = j \sum_{i=1}^{|X|} i \leq k \sum_{i=1}^{|Y|} i.$$

- Special case: If $G = (X, Y, E)$ is regular of degree k and has $2n$ vertices, then $\mu_s(G) = \dfrac{kn(n+1)}{2}$.

Biregular Graphs

- A graph $G = (V, E)$ is biregular if $V = V_1 + V_2$, where V_1 is regular of degree j and V_2 is regular of degree k for some j, k. We call G strictly biregular if $j \neq k$.
- A graph $G = (X, Y, E)$ is bipartite biregular if G is both bipartite and biregular.
- If $G = (X, Y, E)$ is bipartite biregular, and if the degree partition coincides with the partition $V = X + Y$, then $j|X| = k|Y|$, and it follows that if $j \geq k$, then

$$\mu_s(G) = j \sum_{i=1}^{|X|} i \leq k \sum_{i=1}^{|Y|} i.$$

- Special case: If $G = (X, Y, E)$ is regular of degree k and has $2n$ vertices, then $\mu_s(G) = \dfrac{kn(n+1)}{2}$.

Biregular Graphs

- A graph $G = (V, E)$ is biregular if $V = V_1 + V_2$, where V_1 is regular of degree j and V_2 is regular of degree k for some j, k. We call G strictly biregular if $j \neq k$.
- A graph $G = (X, Y, E)$ is bipartite biregular if G is both bipartite and biregular.
- If $G = (X, Y, E)$ is bipartite biregular, and if the degree partition coincides with the partition $V = X + Y$, then $j|X| = k|Y|$, and it follows that if $j \geq k$, then

$$\mu_s(G) = j \sum_{i=1}^{|X|} i \leq k \sum_{i=1}^{|Y|} i.$$

- Special case: If $G = (X, Y, E)$ is regular of degree k and has $2n$ vertices, then $\mu_s(G) = \dfrac{kn(n+1)}{2}$.

Biregular Graphs

- A graph $G = (V, E)$ is biregular if $V = V_1 + V_2$, where V_1 is regular of degree j and V_2 is regular of degree k for some j, k. We call G strictly biregular if $j \neq k$.
- A graph $G = (X, Y, E)$ is bipartite biregular if G is both bipartite and biregular.
- If $G = (X, Y, E)$ is bipartite biregular, and if the degree partition coincides with the partition $V = X + Y$, then $j|X| = k|Y|$, and it follows that if $j \geq k$, then

$$\mu_s(G) = j \sum_{i=1}^{|X|} i \leq k \sum_{i=1}^{|Y|} i.$$

- Special case: If $G = (X, Y, E)$ is regular of degree k and has $2n$ vertices, then $\mu_s(G) = \dfrac{kn(n+1)}{2}$.

Generalized Petersen Graphs

The generalized Petersen Graph $P(n,k)$ for $n \geq 5$ has $V = \{x_i, y_i | 0 \leq i \leq n-1\}$ and $E = \{x_i x_{i+1}, y_i y_{i+k}, x_i y_i\}$, where the subscripts are expressed modulo n. The best-known instance is $P(5,2)$, the Petersen graph. It turns out that $\mu_s(P(5,2)) = 48$, but our focus is on $\mu_s(P(n,k))$ in general. We consider only $k \leq \lfloor (n-1)/2 \rfloor$.

- Two elementary results about $P(n,k)$:
  - $P(n,k)$ is 3-regular.
  - $P(n,k)$ is bipartite if and only if n is even and k is odd.
- By our result on regular bipartite graphs, if n is even and k is odd, then $\mu_s(P(n,k)) = \dfrac{3n(n+1)}{2}$.

Generalized Petersen Graphs

The generalized Petersen Graph $P(n,k)$ for $n \geq 5$ has $V = \{x_i, y_i | 0 \leq i \leq n-1\}$ and $E = \{x_i x_{i+1}, y_i y_{i+k}, x_i y_i\}$, where the subscripts are expressed modulo n. The best-known instance is $P(5,2)$, the Petersen graph. It turns out that $\mu_s(P(5,2)) = 48$, but our focus is on $\mu_s(P(n,k))$ in general. We consider only $k \leq \lfloor (n-1)/2 \rfloor$.

- Two elementary results about $P(n,k)$:
  - $P(n,k)$ is 3-regular.
  - $P(n,k)$ is bipartite if and only if n is even and k is odd.
- By our result on regular bipartite graphs, if n is even and k is odd, then $\mu_s(P(n,k)) = \dfrac{3n(n+1)}{2}$.

Generalized Petersen Graphs

The generalized Petersen Graph $P(n,k)$ for $n \geq 5$ has $V = \{x_i, y_i | 0 \leq i \leq n-1\}$ and $E = \{x_i x_{i+1}, y_i y_{i+k}, x_i y_i\}$, where the subscripts are expressed modulo n. The best-known instance is $P(5,2)$, the Petersen graph. It turns out that $\mu_s(P(5,2)) = 48$, but our focus is on $\mu_s(P(n,k))$ in general. We consider only $k \leq \lfloor (n-1)/2 \rfloor$.

- Two elementary results about $P(n,k)$:
  - $P(n,k)$ is 3-regular.
  - $P(n,k)$ is bipartite if and only if n is even and k is odd.
- By our result on regular bipartite graphs, if n is even and k is odd, then $\mu_s(P(n,k)) = \dfrac{3n(n+1)}{2}$.

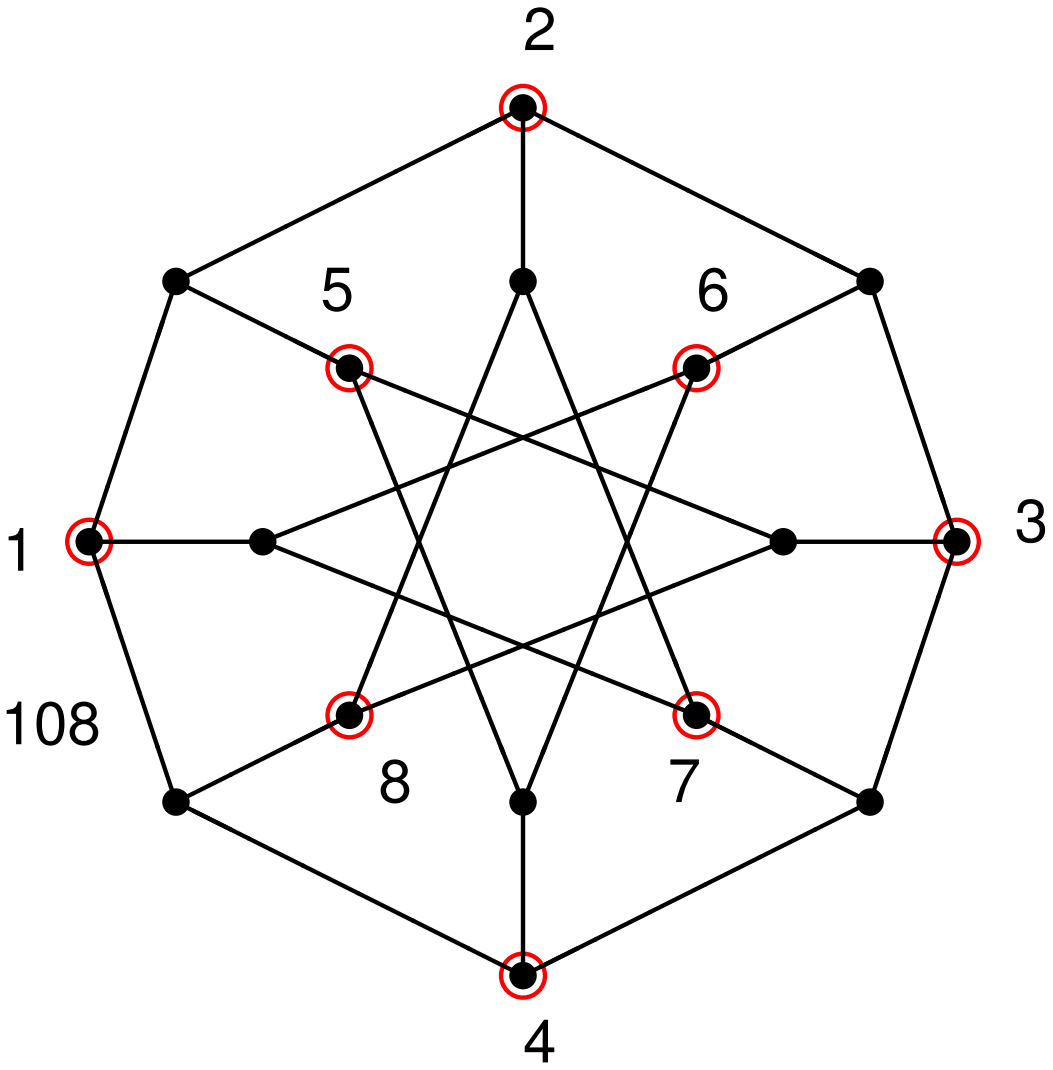
Generalized Petersen Graphs

The generalized Petersen Graph $P(n,k)$ for $n \geq 5$ has $V = \{x_i, y_i | 0 \leq i \leq n-1\}$ and $E = \{x_i x_{i+1}, y_i y_{i+k}, x_i y_i\}$, where the subscripts are expressed modulo n. The best-known instance is $P(5,2)$, the Petersen graph. It turns out that $\mu_s(P(5,2)) = 48$, but our focus is on $\mu_s(P(n,k))$ in general. We consider only $k \leq \lfloor (n-1)/2 \rfloor$.

- Two elementary results about $P(n,k)$:
  - $P(n,k)$ is 3-regular.
  - $P(n,k)$ is bipartite if and only if n is even and k is odd.
- By our result on regular bipartite graphs, if n is even and k is odd, then $\mu_s(P(n,k)) = \dfrac{3n(n+1)}{2}$.

Example: $P(8,3)$

$P(8,3))$:



$$\mu_s(P(8,3)) = 108$$

$$= \frac{3 \cdot 8 \cdot 9}{2}$$

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

Perfection

- The generalized Petersen graphs, and some others that we looked at during many discussions of the msvc problem, are interesting in their own right. Does it make sense to break ground on a new problem and on an unfamiliar class of graphs simultaneously?
- In search of a well-understood family of graphs, rich in structure, why look further than subclasses of the perfect graphs?
- A graph G is perfect if $\chi(H) = \omega(H)$ for all induced subgraphs $H \subseteq G$.
- Many problems that are NP-complete in general are quite easily solved on numerous classes of perfect graphs, so it seems natural to look at perfect graphs in the context of mscv.

Perfection

- The generalized Petersen graphs, and some others that we looked at during many discussions of the msvc problem, are interesting in their own right. Does it make sense to break ground on a new problem and on an unfamiliar class of graphs simultaneously?
- In search of a well-understood family of graphs, rich in structure, why look further than subclasses of the perfect graphs?
- A graph G is perfect if $\chi(H) = \omega(H)$ for all induced subgraphs $H \subseteq G$.
- Many problems that are NP-complete in general are quite easily solved on numerous classes of perfect graphs, so it seems natural to look at perfect graphs in the context of mscv.

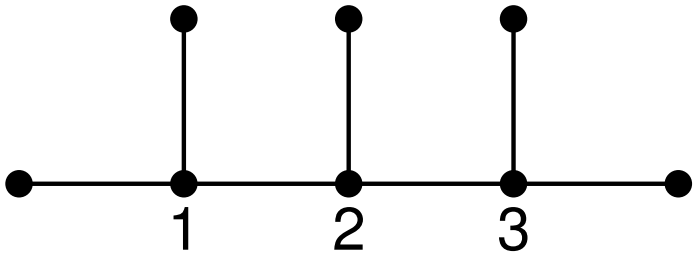
Perfection

- The generalized Petersen graphs, and some others that we looked at during many discussions of the msvc problem, are interesting in their own right. Does it make sense to break ground on a new problem and on an unfamiliar class of graphs simultaneously?
- In search of a well-understood family of graphs, rich in structure, why look further than subclasses of the perfect graphs?
- A graph G is perfect if $\chi(H) = \omega(H)$ for all induced subgraphs $H \subseteq G$.
- Many problems that are NP-complete in general are quite easily solved on numerous classes of perfect graphs, so it seems natural to look at perfect graphs in the context of mscv.

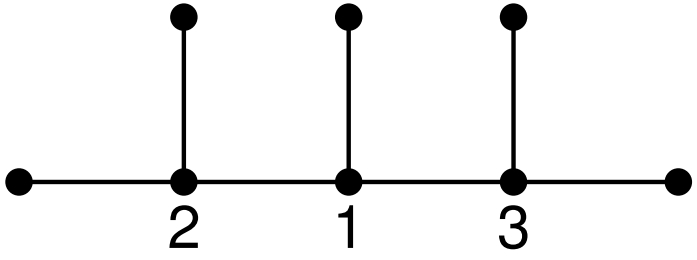
Perfection

- The generalized Petersen graphs, and some others that we looked at during many discussions of the msvc problem, are interesting in their own right. Does it make sense to break ground on a new problem and on an unfamiliar class of graphs simultaneously?
- In search of a well-understood family of graphs, rich in structure, why look further than subclasses of the perfect graphs?
- A graph G is perfect if $\chi(H) = \omega(H)$ for all induced subgraphs $H \subseteq G$.
- Many problems that are NP-complete in general are quite easily solved on numerous classes of perfect graphs, so it seems natural to look at perfect graphs in the context of mscv.

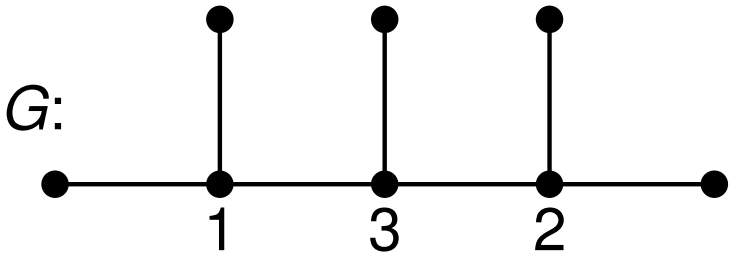
We already have results for paths, which are bipartite and therefore perfect. What about caterpillars? We begin with biregular caterpillars. With three consecutive vertices of maximum degree, a complication arises. Two of the possible labelings agree, but are not optimal.



$$\sum_e g(e) = 13.$$

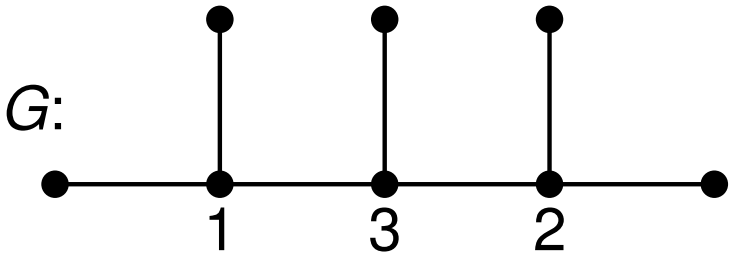


$$\sum_e g(e) = 13.$$



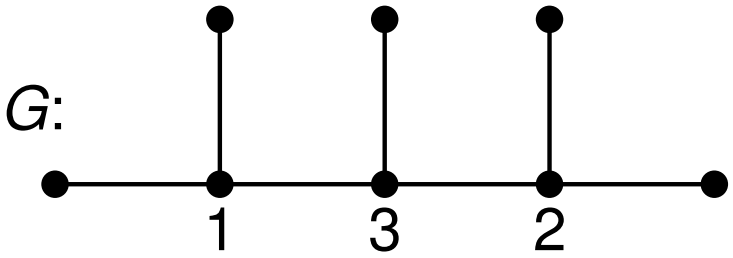
$$\sum_e g(e) = 12$$
$$= \mu_s(G).$$

- In one of the suboptimal labelings, we prematurely labeled a vertex of maximum degree that was adjacent to two others that also had maximum degree. In the other, we assigned the next available label to a vertex of maximum degree that already had its degree effectively reduced in the previous step.
- How do we avoid this?
- Enter the interval graphs.



$$\sum_e g(e) = 12$$
$$= \mu_s(G).$$

- In one of the suboptimal labelings, we prematurely labeled a vertex of maximum degree that was adjacent to two others that also had maximum degree. In the other, we assigned the next available label to a vertex of maximum degree that already had its degree effectively reduced in the previous step.
- How do we avoid this?
- Enter the interval graphs.

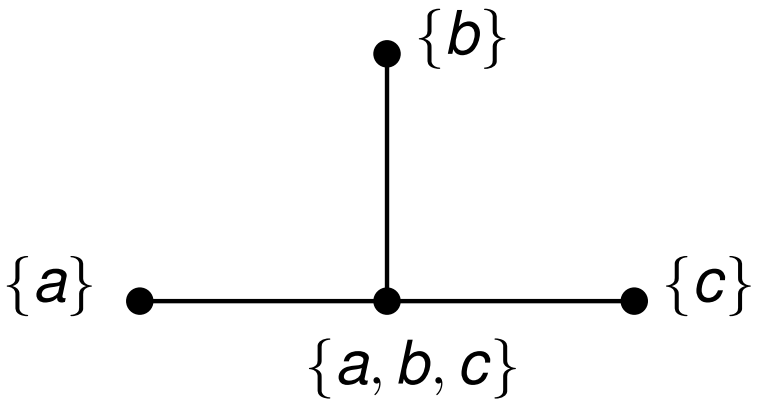


$$\sum_e g(e) = 12$$
$$= \mu_s(G).$$

- In one of the suboptimal labelings, we prematurely labeled a vertex of maximum degree that was adjacent to two others that also had maximum degree. In the other, we assigned the next available label to a vertex of maximum degree that already had its degree effectively reduced in the previous step.
- How do we avoid this?
- Enter the interval graphs.

Intersection Graphs

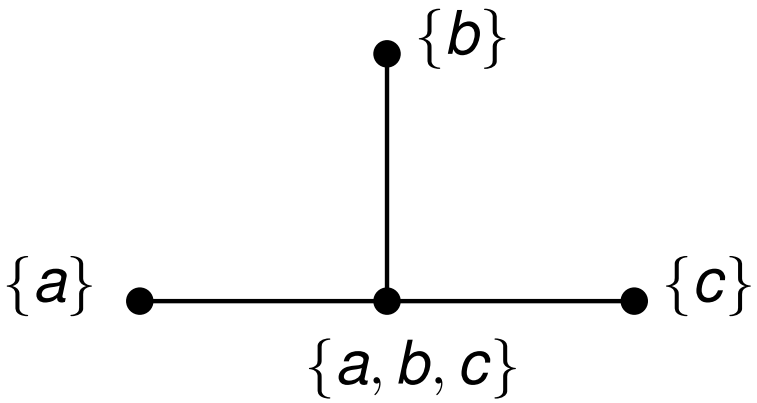
- A graph $G = (V, E)$ is an intersection graph if the vertices can be labeled as sets, with an edge between vertices X and Y iff $X \cap Y \neq \emptyset$.



- An interval graph is the intersection graph of a family of intervals on the real line.

Intersection Graphs

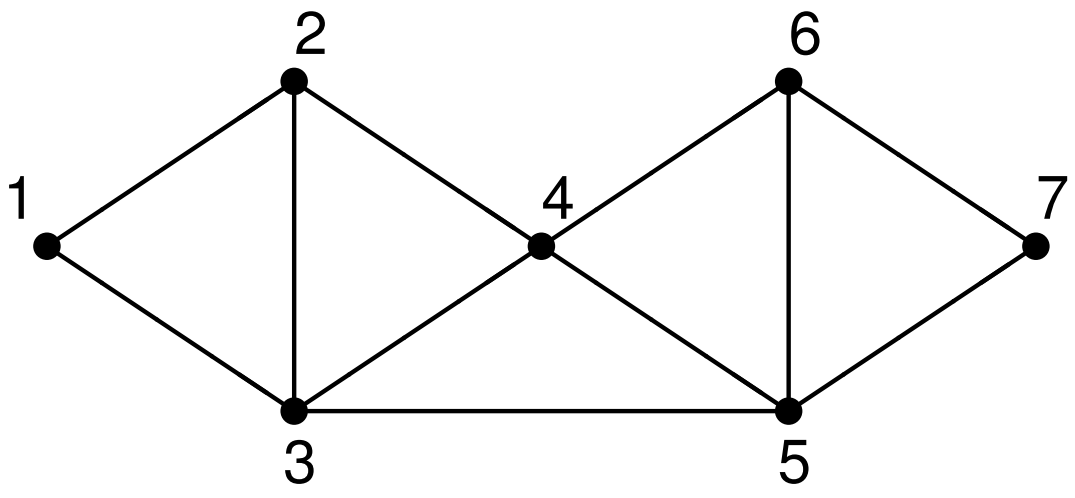
- A graph $G = (V, E)$ is an intersection graph if the vertices can be labeled as sets, with an edge between vertices X and Y iff $X \cap Y \neq \emptyset$.



- An interval graph is the intersection graph of a family of intervals on the real line.

Interval Elimination Orderings

- Every interval graph can be assigned an interval elimination ordering, which is a labeling $v_1, v_2, \ldots, v_n$, where the labels in $N^-(v_i) = \{v_j | j = i \text{ or } v_i v_j \in E\}$ occur consecutively [Laskar & Jamison].



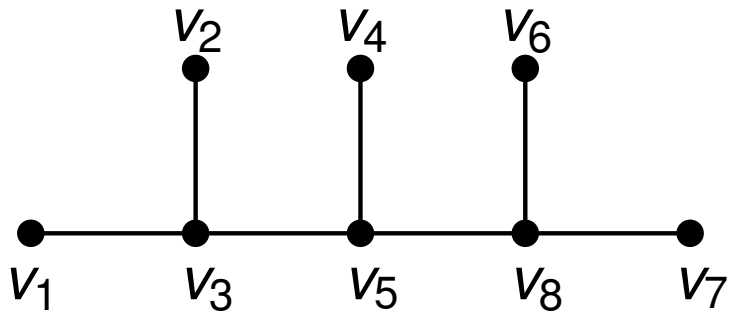
An Algorithm for Biregular Caterpillars

- We now can give an algorithm for finding the msvc number of a biregular caterpillar T.
- We start by constructing an IEO for T. Among all vertices of maximum degree, choose the vertex, say v, with smallest index in the IEO. Label this vertex with the smallest available label (initially 1). Now apply the same procedure recursively to $T - \{v\}$.

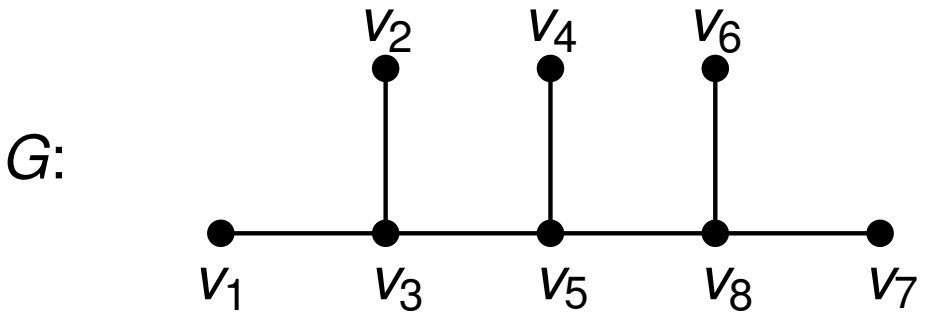
An Algorithm for Biregular Caterpillars

- We now can give an algorithm for finding the msvc number of a biregular caterpillar T.
- We start by constructing an IEO for T. Among all vertices of maximum degree, choose the vertex, say v, with smallest index in the IEO. Label this vertex with the smallest available label (initially 1). Now apply the same procedure recursively to $T - \{v\}$.

G:



- A caterpillar G is an interval tree, so we can construct an IEO for G.
- By choosing a vertex of maximum degree with smallest label in the IEO, we avoid the "betweenness" problem.



- A caterpillar G is an interval tree, so we can construct an IEO for G.
- By choosing a vertex of maximum degree with smallest label in the IEO, we avoid the “betweenness” problem.

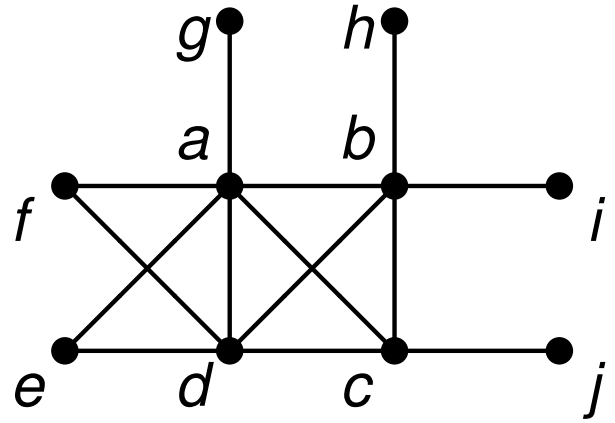
Caterpillars in General

- Armed with the preceding result, we have a procedure for arbitrary caterpillars.

Split Graphs

- A split graph is a graph $G = (V, E)$ for which V can be partitioned as $V = V_1 + V_2$, where $G[V_1]$ is a clique and $G[V_2]$ is an independent set. Split graphs are perfect.
- Here $V_1 = \{a, b, c, d\}$ and $V_2 = \{e, f, g, h, i, j\}$.

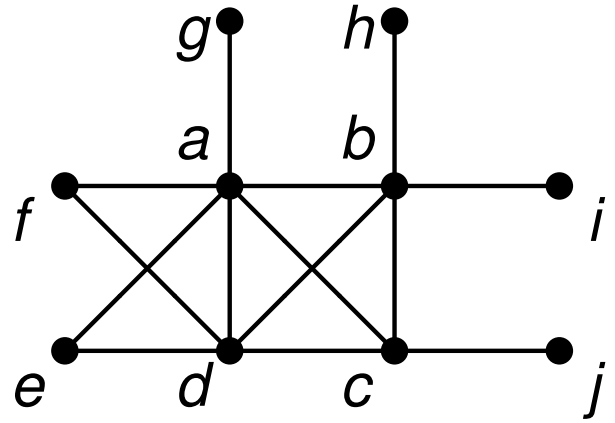
H:



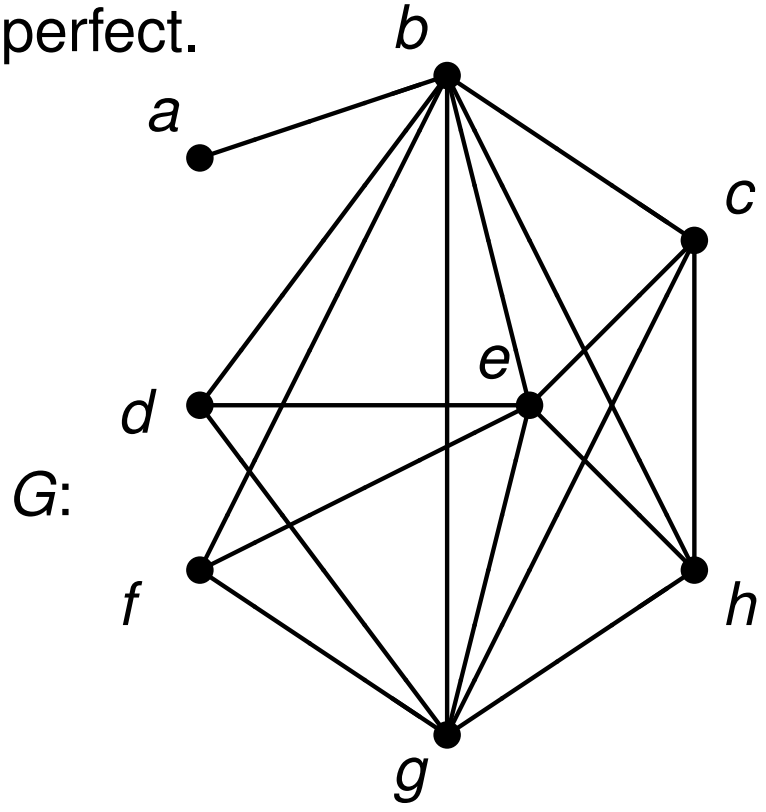
Split Graphs

- A split graph is a graph $G = (V, E)$ for which V can be partitioned as $V = V_1 + V_2$, where $G[V_1]$ is a clique and $G[V_2]$ is an independent set. Split graphs are perfect.
- Here $V_1 = \{a, b, c, d\}$ and $V_2 = \{e, f, g, h, i, j\}$.

H:



- A threshold graph is a highly structured split graph $G = (V, E)$ for which several characterizations exist. Threshold graphs are perfect.



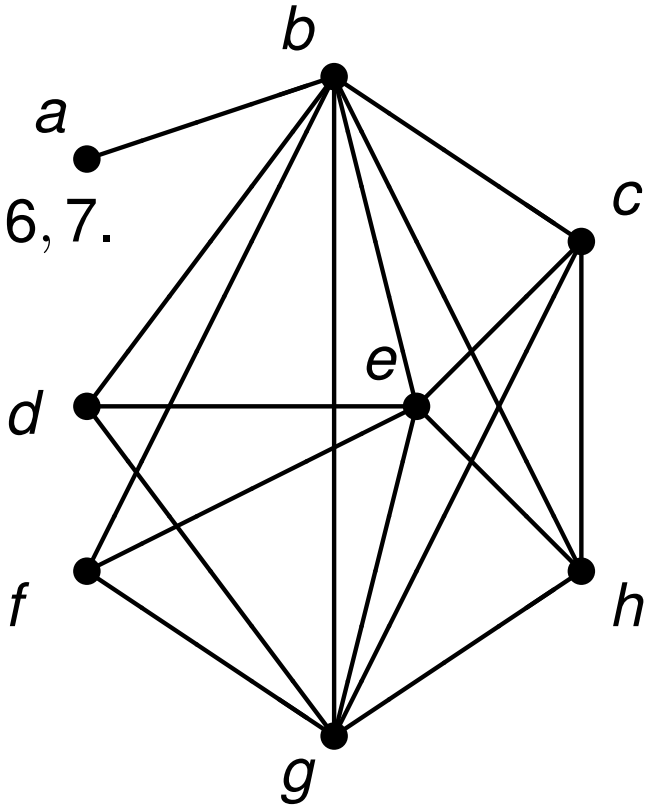
A Useful Characterization of Threshold Graphs

- The vertex set of a threshold graph with distinct positive degrees $d_1 \leq d_2 \leq \cdots \leq d_m$ can be partitioned as $V = V_1 + V_2 + \cdots + V_m$, where V_i contains all vertices of degree d_i. For $x \in V_i$ and $y \in V_j$, $xy \in E$ iff $i + j > m$.
- Threshold graphs are split graphs.

Vertex Degrees are $1, 2, 4, 6, 7$.

$V_5 = \{b\}$
$V_4 = \{e, g\}$
$V_3 = \{c, h\}$
$V_2 = \{d, f\}$
$V_1 = \{a\}$

G:



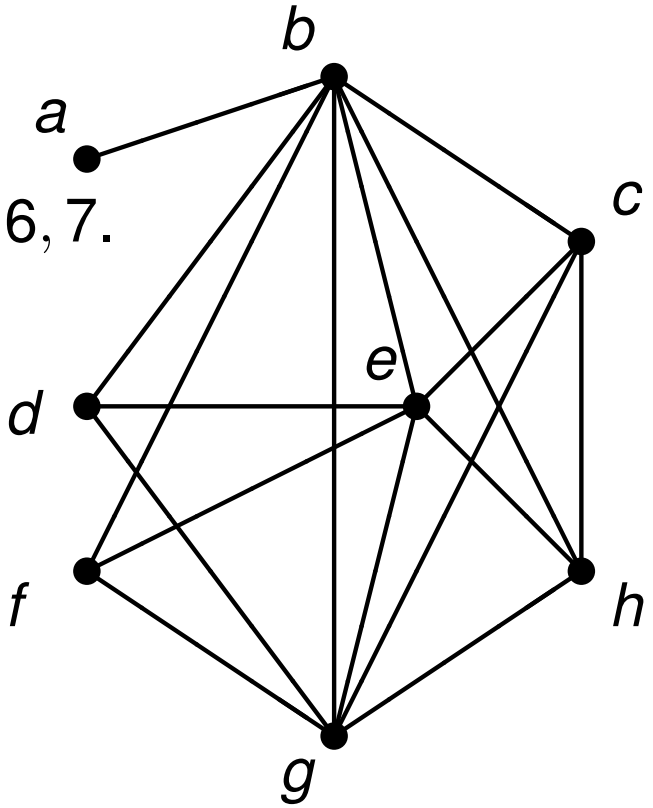
A Useful Characterization of Threshold Graphs

- The vertex set of a threshold graph with distinct positive degrees $d_1 \leq d_2 \leq \cdots \leq d_m$ can be partitioned as $V = V_1 + V_2 + \cdots + V_m$, where V_i contains all vertices of degree d_i. For $x \in V_i$ and $y \in V_j$, $xy \in E$ iff $i + j > m$.
- Threshold graphs are split graphs.

Vertex Degrees are $1, 2, 4, 6, 7$.

$V_5 = \{b\}$
$V_4 = \{e, g\}$
$V_3 = \{c, h\}$
$V_2 = \{d, f\}$
$V_1 = \{a\}$

G:



An Algorithm for Split Graphs

- We can compute the msvc number of a split graph in polynomial time.
- The msvc number of a split graph is a function of its degree sequence.

Theorem

Let $G = (V, E)$ be a split graph on n vertices. Let $V = \{v_1, v_2, \ldots, v_n\}$, and without loss of generality assume that $i < j \rightarrow \deg(v_i) \geq \deg(v_j)$. Let f be the labeling of V defined by $f(v_i) = i$. Then f is an msvc labeling of G.

An Algorithm for Split Graphs

- We can compute the msvc number of a split graph in polynomial time.
- The msvc number of a split graph is a function of its degree sequence.

Theorem

Let $G = (V, E)$ be a split graph on n vertices. Let $V = \{v_1, v_2, \ldots, v_n\}$, and without loss of generality assume that $i < j \rightarrow \deg(v_i) \geq \deg(v_j)$. Let f be the labeling of V defined by $f(v_i) = i$. Then f is an msvc labeling of G.

An Algorithm for Split Graphs

- We can compute the msvc number of a split graph in polynomial time.
- The msvc number of a split graph is a function of its degree sequence.

Theorem

Let $G = (V, E)$ be a split graph on n vertices. Let $V = \{v_1, v_2, \ldots, v_n\}$, and without loss of generality assume that $i < j \rightarrow \deg(v_i) \geq \deg(v_j)$. Let f be the labeling of V defined by $f(v_i) = i$. Then f is an msvc labeling of G.

An Algorithm for Split Graphs

- The proof of the theorem is simple. The split property is hereditary, so it suffices to show that any labeling that assigns 1 to a vertex of maximum degree achieves a better result than any labeling that assigns 1 to a vertex of smaller degree.
- The algorithm is simple, too:
  - Sort the vertices in nonincreasing order by degree.
  - Iterate: Assign the smallest available label to the first available vertex. Update the degree sequence.
  - When the next degree in the sequence is zero, stop.

An Algorithm for Split Graphs

- The proof of the theorem is simple. The split property is hereditary, so it suffices to show that any labeling that assigns 1 to a vertex of maximum degree achieves a better result than any labeling that assigns 1 to a vertex of smaller degree.
- The algorithm is simple, too:
  - Sort the vertices in nonincreasing order by degree.
  - Iterate: Assign the smallest available label to the first available vertex. Update the degree sequence.
  - When the next degree in the sequence is zero, stop.

An Algorithm for Split Graphs

- The proof of the theorem is simple. The split property is hereditary, so it suffices to show that any labeling that assigns 1 to a vertex of maximum degree achieves a better result than any labeling that assigns 1 to a vertex of smaller degree.
- The algorithm is simple, too:
  - Sort the vertices in nonincreasing order by degree.
  - Iterate: Assign the smallest available label to the first available vertex. Update the degree sequence.
  - When the next degree in the sequence is zero, stop.

An Algorithm for Split Graphs

- The proof of the theorem is simple. The split property is hereditary, so it suffices to show that any labeling that assigns 1 to a vertex of maximum degree achieves a better result than any labeling that assigns 1 to a vertex of smaller degree.
- The algorithm is simple, too:
  - Sort the vertices in nonincreasing order by degree.
  - Iterate: Assign the smallest available label to the first available vertex. Update the degree sequence.
  - When the next degree in the sequence is zero, stop.

An Algorithm for Split Graphs

- The proof of the theorem is simple. The split property is hereditary, so it suffices to show that any labeling that assigns 1 to a vertex of maximum degree achieves a better result than any labeling that assigns 1 to a vertex of smaller degree.
- The algorithm is simple, too:
  - Sort the vertices in nonincreasing order by degree.
  - Iterate: Assign the smallest available label to the first available vertex. Update the degree sequence.
  - When the next degree in the sequence is zero, stop.

A Formula for the msvc Number of a Split Graph

- As an immediate corollary, we have a formula for the msvc number of a split graph in terms of its degree sequence.

Corollary

Let $G = (V, E)$ be a split graph. Let $d_1 \geq d_2 \geq \cdots \geq d_{|V|}$ be the degree sequence of G. Let $k = \max\{i | d_i - i + 1 > 0\}$. Then the msvc number of G is

$$\mu_s(G) = \sum_{i=1}^{k} i(d_i - i + 1).$$

A Formula for the msvc Number of a Split Graph

- As an immediate corollary, we have a formula for the msvc number of a split graph in terms of its degree sequence.

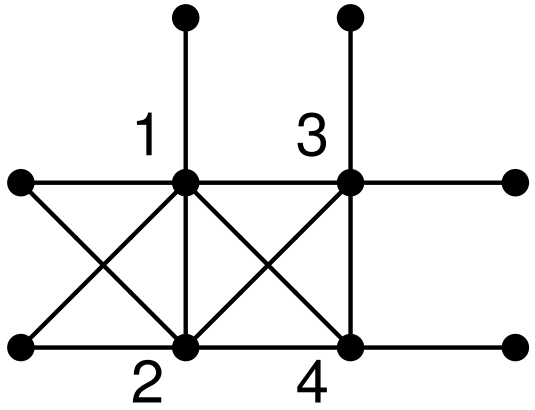
Corollary

Let $G = (V, E)$ be a split graph. Let $d_1 \geq d_2 \geq \cdots \geq d_{|V|}$ be the degree sequence of G. Let $k = \max\{i | d_i - i + 1 > 0\}$. Then the msvc number of G is

$$\mu_s(G) = \sum_{i=1}^{k} i(d_i - i + 1).$$

An msvc Labeling of H

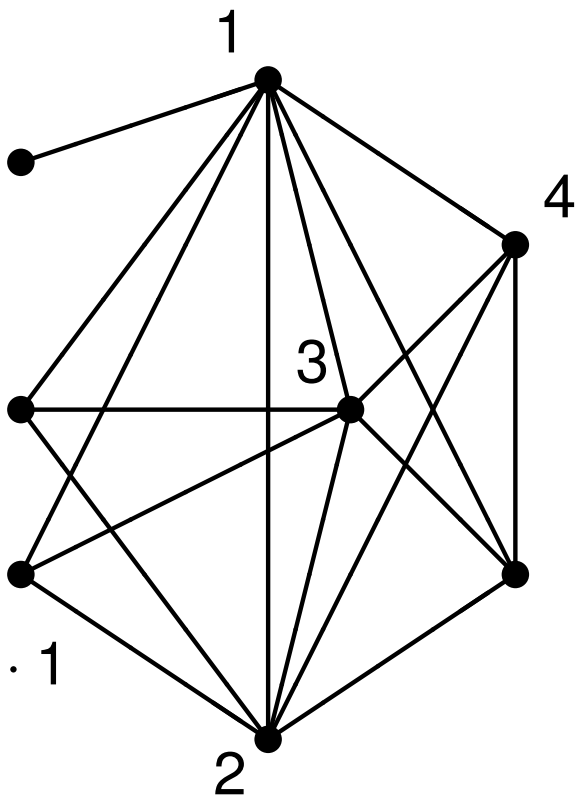
H:



$$\begin{aligned}\mu_s(H) &= 1 \cdot 6 + 2 \cdot 4 + 3 \cdot 3 + 4 \cdot 1 \\ &= 27.\end{aligned}$$

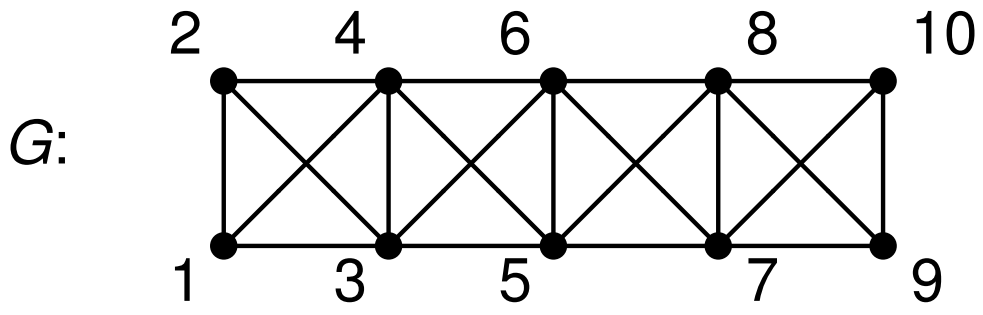
An msvc Labeling of G

G:

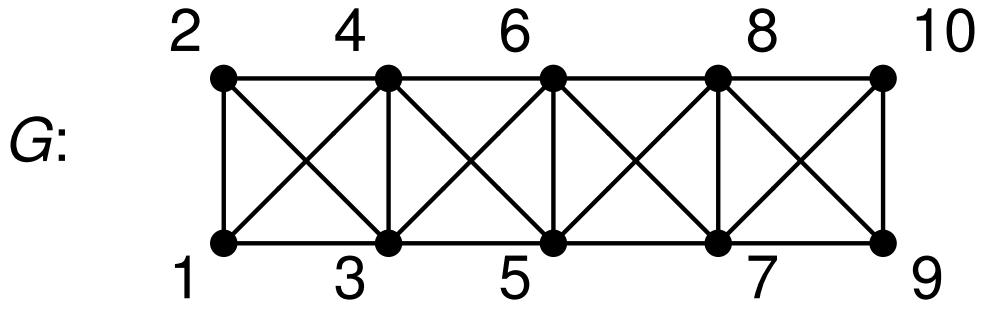


$$\mu_s(G) = 1 \cdot 7 + 2 \cdot 5 + 3 \cdot 4 + 4 \cdot 1$$
$$= 33.$$

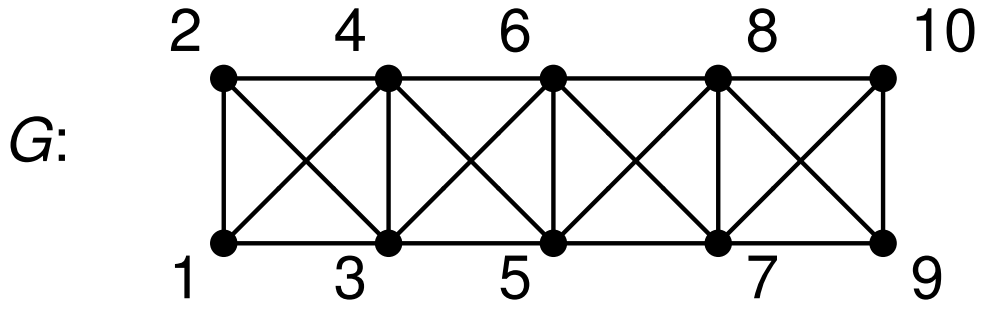
Unit Interval Graphs



- A graph G is unit interval if it is the intersection of a family of unit-length intervals on the real line. The interval elimination ordering of G is an interval elimination ordering in both directions: if we interchange labels i and $n - i$, we have another interval elimination ordering (called bicompatible). This existence of a BEO is characteristic of unit interval graphs [Laskar & Jamison].



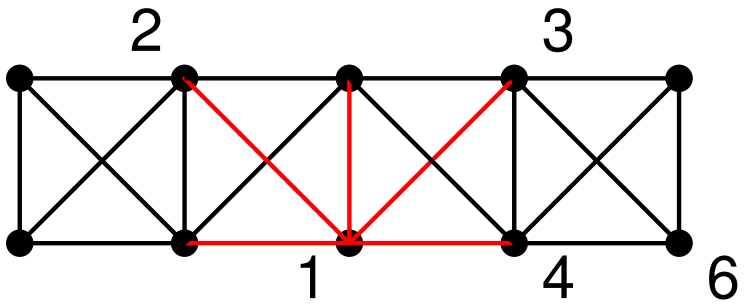
- A vertex v is simplicial if $N[v]$ is a clique. Just as the focus in caterpillars is on the internal vertices, the focus in UI graphs is on the nonsimplicial vertices.
- This graph has six nonsimplicial vertices, labeled $3, 4, 5, 6, 7, 8$. All have degree 5. Vertices 5 and 6 are in the middle.



- A vertex v is simplicial if $N[v]$ is a clique. Just as the focus in caterpillars is on the internal vertices, the focus in UI graphs is on the nonsimplicial vertices.
- This graph has six nonsimplicial vertices, labeled $3, 4, 5, 6, 7, 8$. All have degree 5. Vertices 5 and 6 are in the middle.

G:

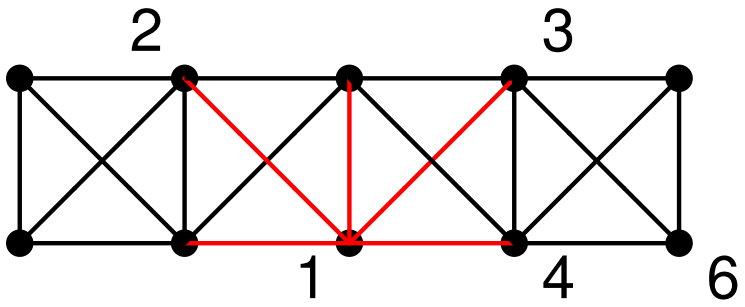
$\text{Cost}(f) = 65$



- If we assign the least label to vertex 5 (or 6), we decrease the potential of all of the remaining five nonsimplicial vertices.
- No matter what we do next, we end up with a suboptimal result.

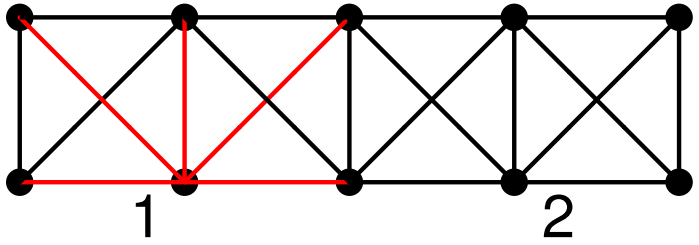
G:

$\text{Cost}(f) = 65$



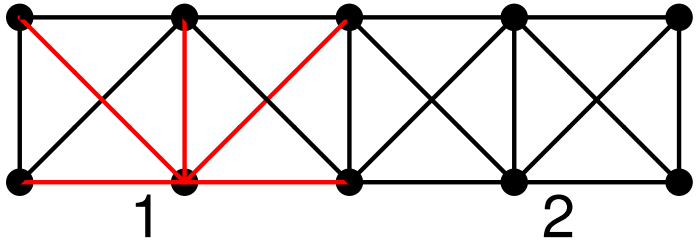
- If we assign the least label to vertex 5 (or 6), we decrease the potential of all of the remaining five nonsimplicial vertices.
- No matter what we do next, we end up with a suboptimal result.

G:



- If we instead assign the least label to the first nonsimplicial vertex in the BEO, we decrease the potential of only three of the remaining nonsimplicial vertices.
- The search for a vertex of maximum degree now leads to a nonsimplicial vertex that was unaffected by the first assignment.

G:

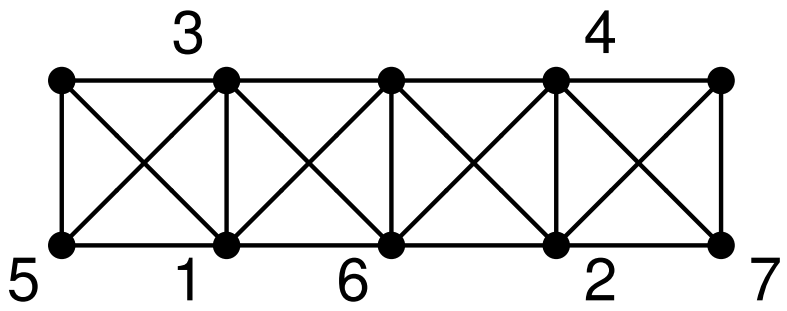


- If we instead assign the least label to the first nonsimplicial vertex in the BEO, we decrease the potential of only three of the remaining nonsimplicial vertices.
- The search for a vertex of maximum degree now leads to a nonsimplicial vertex that was unaffected by the first assignment.

An Optimal Labeling

G:

$\mu_s(G) = 61$



- Continuing in this fashion, guided by the elimination ordering, we attain an msvc labeling.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Summing Up

- The known classes of perfect graphs are rich in structure and attributes that assist in obtaining results. Many have one or more of the following:
  - Forbidden subgraph characterizations.
  - Elimination orderings.
  - Characterizations as intersection graphs of well-defined families of sets.
  - Characterizations in terms of vertex degrees.
  - Polyhedral characterizations.
- Many have nice algorithmic properties, enabling us to apply a constructive approach.

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

What's next?

- Interval graphs in general are only slightly more complicated in structure than unit interval graphs and caterpillars.
- There are several hundred known classes of perfect graphs. Many of these might allow polynomial-time algorithms and/or closed-form expressions for computing the msvc number.
- The generalized Petersen graphs are still out there.

What's next?

- Interval graphs in general are only slightly more complicated in structure than unit interval graphs and caterpillars.
- There are several hundred known classes of perfect graphs. Many of these might allow polynomial-time algorithms and/or closed-form expressions for computing the msvc number.
- The generalized Petersen graphs are still out there.

What's next?

- Interval graphs in general are only slightly more complicated in structure than unit interval graphs and caterpillars.
- There are several hundred known classes of perfect graphs. Many of these might allow polynomial-time algorithms and/or closed-form expressions for computing the msvc number.
- The generalized Petersen graphs are still out there.

Outline

1. Intro
   - Definitions
   - Motivation: Data Migration
   - Some Examples
2. Phase II
   - Regular Graphs
   - Biregular Graphs
   - Generalized Petersen Graphs
3. Back to Basics
   - Perfect Graphs
   - Threshold and Split Graphs
4. Looking Ahead
5. References

References

1. U. Feige, L. Lovász, P. Tetali, *Approximating Min-Sum Set Cover*, Algorithmica 40 (2004), 219–234.
2. R. Gera, S. Horton, C. Rasmussen, P. Stănică, *Results on the Min-Sum Vertex Cover Problem*, to appear.
3. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press 1980.
4. Y. Kim, *Data Migration to Minimize the Average Completion Time*, Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, 97–98.
5. R. Laskar, R. E. Jamison, *Elimination Orderings of Chordal Graphs*, Proc. Indian Statistics Institute, 1982.