Listing all fixed-length simple cycles in sparse graphs in optimal time

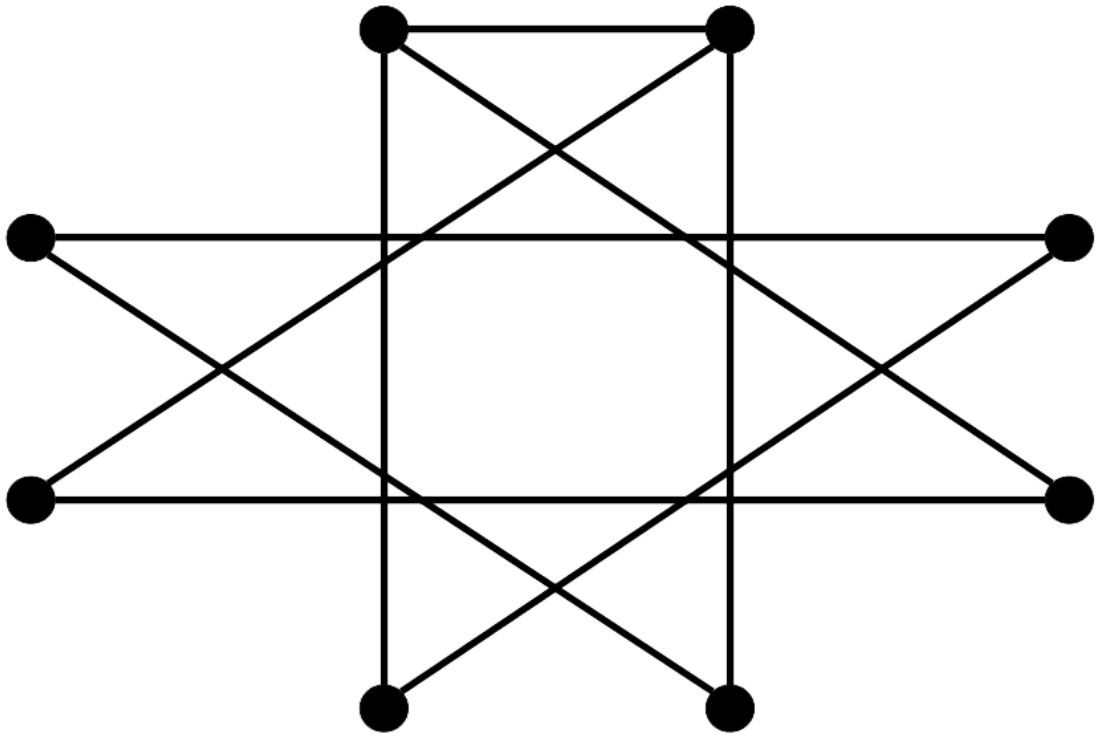
George Manoussakis

Université Paris Saclay, Université Paris Sud, LRI-CNRS, France

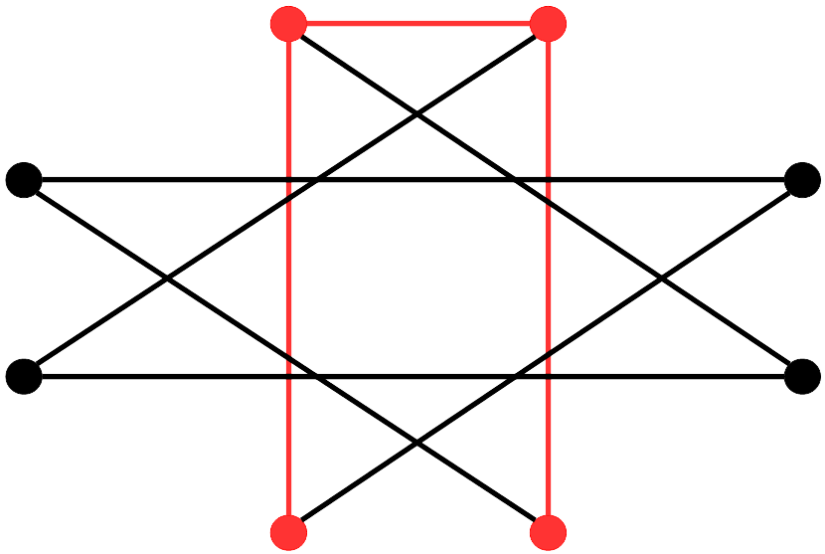
Introduction

Input object: *a graph*

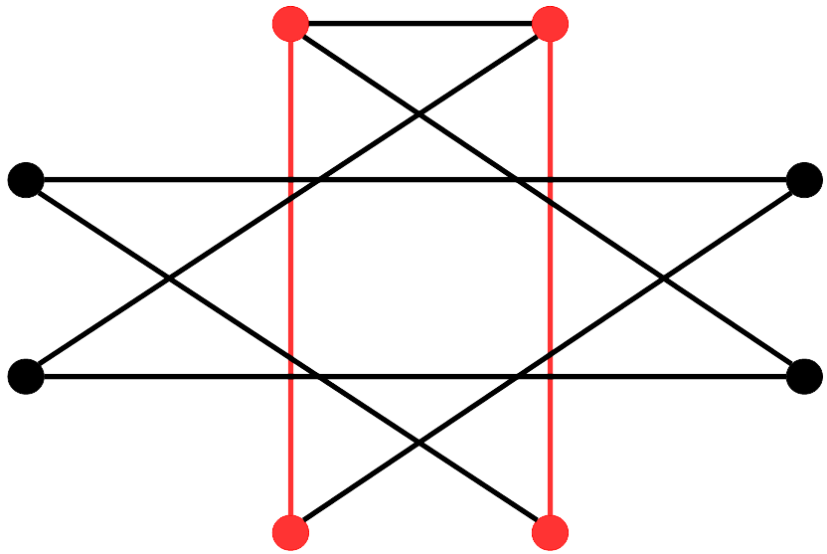
$G = (V, E)$ a graph: $|V| = n$ vertices and $|E| = m$ edges.



Question: find all (induced) subgraphs with some property

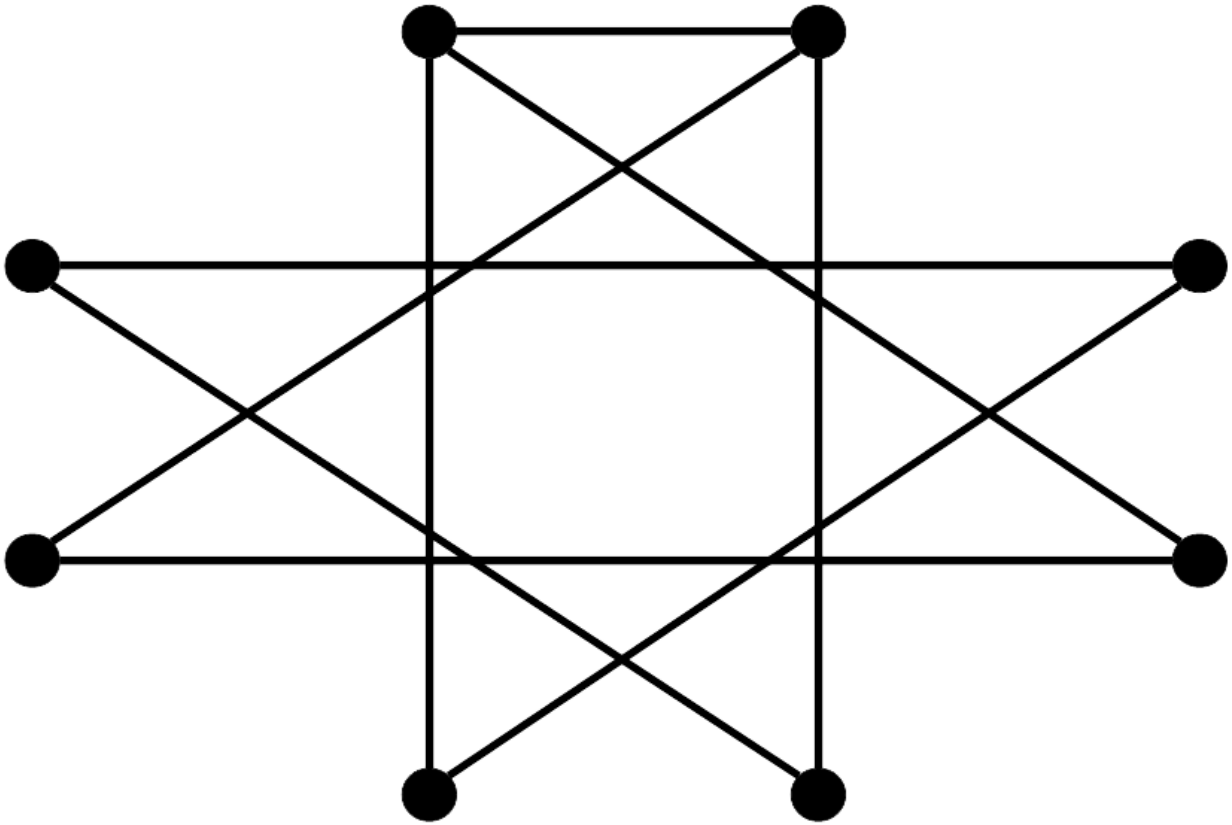


induced subgraph

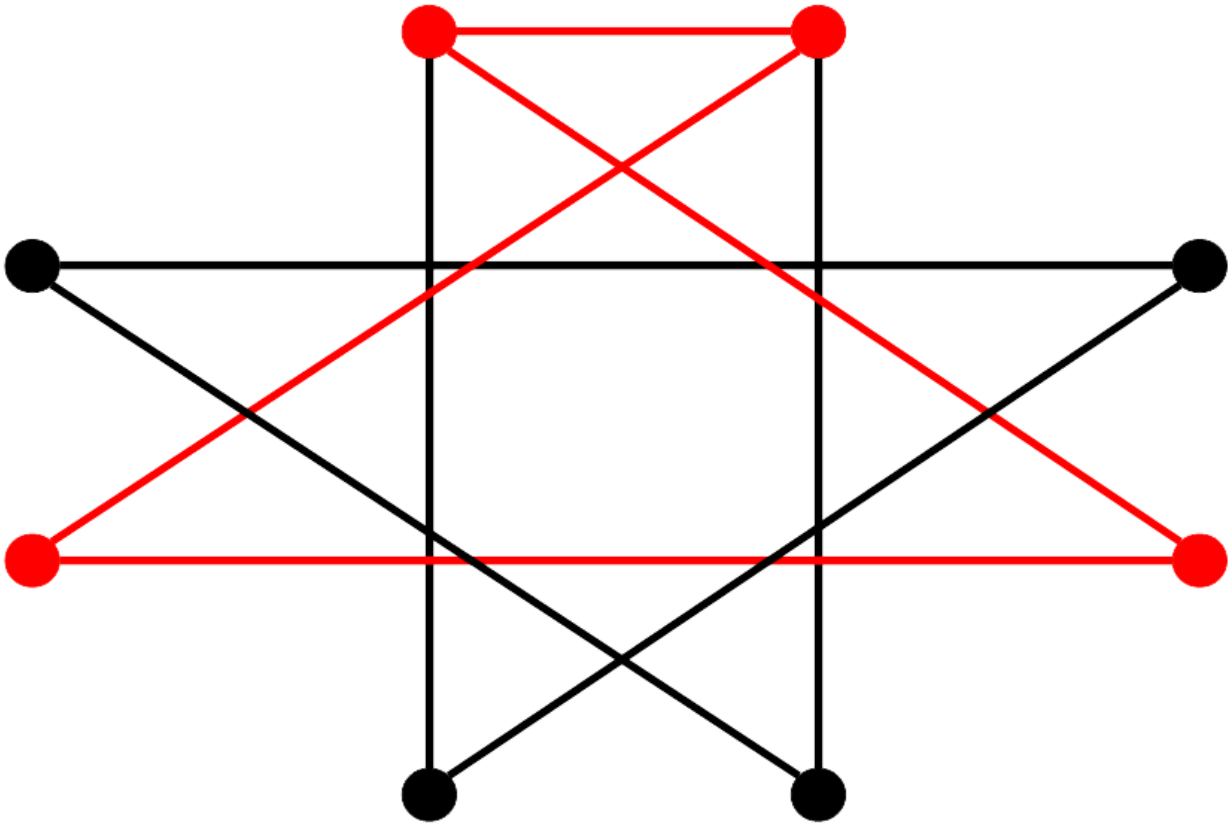


subgraph

Example: find all cycles of length four



Example: find all cycles of length four



Introduction - k-degenerate graphs

Definition (k-degenerate graphs)

Graph k-degenerate if k is smallest integer such that:

- every induced subgraph has a vertex of degree at most k,
  or equivalently

Introduction - k-degenerate graphs

Definition (k-degenerate graphs)

Graph k-degenerate if k is smallest integer such that:

- every induced subgraph has a vertex of degree at most k, **or equivalently**
- it has *acyclic orientation* in which every vertex has out-degree bounded by k

Introduction - k-degenerate graphs

Definition (k-degenerate graphs)

Graph k-degenerate if k is smallest integer such that:

- every induced subgraph has a vertex of degree at most k, **or equivalently**
- it has *acyclic orientation* in which every vertex has out-degree bounded by k

Example:

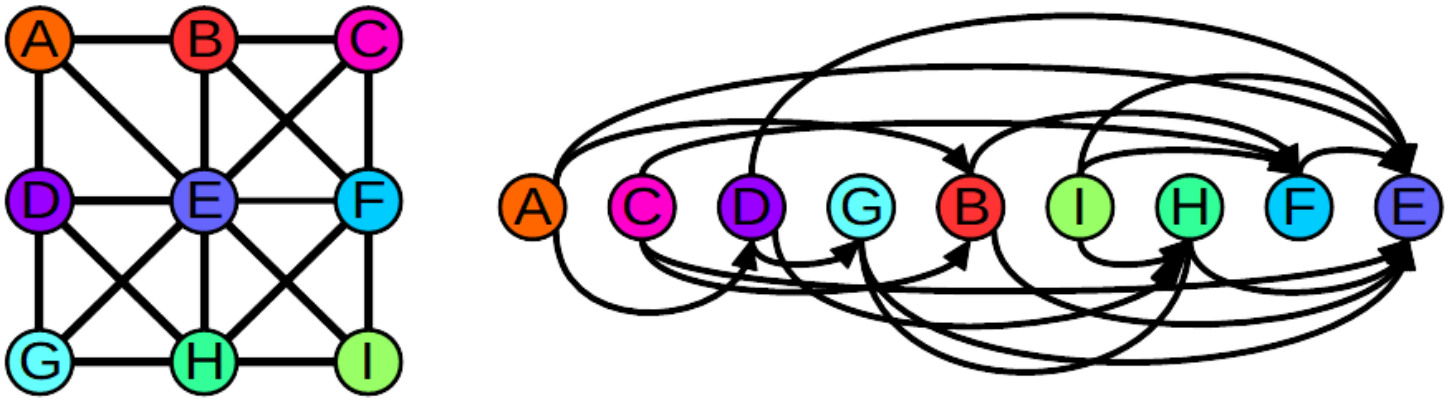


Figure: A 3-degenerate graph and a 3-bounded orientation

Introduction - k-degenerate graphs

Why degenerate graphs

Introduction - k-degenerate graphs

Why degenerate graphs

- generalize many graphs: planar graphs 5-degenerate, trees 1-degenerate, Barabási-Albert graphs etc.

Introduction - k-degenerate graphs

Why degenerate graphs

- generalize many graphs: planar graphs 5-degenerate, trees 1-degenerate, Barabási-Albert graphs etc.
- many real world graphs have low degeneracy

Graph	n	m	k
citationCiteseer	268,495	1.1M	15
Flickr	1.72M	15.6M	568
uk-2002	18M	261M	943
Twitter	41.7M	1.20B	2490

Introduction - k-degenerate graphs

Why degenerate graphs

- generalize many graphs: planar graphs 5-degenerate, trees 1-degenerate, Barabási-Albert graphs etc.
- many real world graphs have low degeneracy

Graph	n	m	k
citationCiteseer	268,495	1.1M	15
Flickr	1.72M	15.6M	568
uk-2002	18M	261M	943
Twitter	41.7M	1.20B	2490

theoretical/practical

Input: graph G, $p \in \mathbb{N}$
Output: *all cycles subgraphs* of G with p vertices (without duplicates)

Input: graph G, $p \in \mathbb{N}$
Output: *all cycles subgraphs* of G with p vertices (without duplicates)

This talk: $\mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$

Input: graph G, $p \in \mathbb{N}$
Output: *all cycles subgraphs* of G with p vertices (without duplicates)

This talk: $\mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$

worst case output size optimal:

$\forall k, p$ and $n \geq kp, \exists$ n-order k-degenerate graph with $\Omega(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$ p-cycles.

Input: graph G, $p \in \mathbb{N}$
Output: *all cycles subgraphs* of G with p vertices (without duplicates)

This talk: $\mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$

worst case output size optimal:

$\forall k, p$ and $n \geq kp, \exists$ n-order k-degenerate graph with $\Omega(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$ p-cycles.

space optimal:

only requires memory needed to store the graph

Related work:

1 cycle:

Related work:

1 cycle:

- C_3 in planar graphs in linear time [Papadimitriou *et al.* '81]

Related work:

1 cycle:

- C_3 in planar graphs in linear time [Papadimitriou *et al.* '81]
- C_3 and C_4 in planar and degenerate in linear time [Chiba *et al.* '85], [Chrobak *et al.* '91]

Related work:

1 cycle:

- C_3 in planar graphs in linear time [Papadimitriou *et al.* '81]
- C_3 and C_4 in planar and degenerate in linear time [Chiba *et al.* '85], [Chrobak *et al.* '91]
- C_5 and C_6 in planar in $\mathcal{O}(n \log n)$ [Richardson *et al.* '86]

Related work:

1 cycle:

- C_3 in planar graphs in linear time [Papadimitriou *et al.* '81]
- C_3 and C_4 in planar and degenerate in linear time
  [Chiba *et al.* '85], [Chrobak *et al.* '91]
- C_5 and C_6 in planar in $\mathcal{O}(n \log n)$
  [Richardson *et al.* '86]
- Any length general and degenerate graphs
  [Alon *et al.* '97]
  for induced cycles : [Cai *et al.* '06]

Related work:

all cycles fixed length:

Related work:

all cycles fixed length:

- any length in planar graphs in $\mathcal{O}(occurences)$ [Eppstein '95]

Related work:

all cycles fixed length:

- any length in planar graphs in $\mathcal{O}(occurences)$ [Eppstein '95]
- cycles of length 4 and 5 in degenerate graphs in $\mathcal{O}(occurences)$ [Kowalik '03]

Related work:

all cycles fixed length:

- any length in planar graphs in $\mathcal{O}(occurences)$ [Eppstein '95]
- cycles of length 4 and 5 in degenerate graphs in $\mathcal{O}(occurences)$ [Kowalik '03]
- any length in general graphs, randomized in $\mathcal{O}(f * occurences)$ with f time to find one occurence [Meeks '16].

Related work:

all cycles fixed length:

- any length in planar graphs in $\mathcal{O}(occurences)$ [Eppstein '95]
- cycles of length 4 and 5 in degenerate graphs in $\mathcal{O}(occurences)$ [Kowalik '03]
- any length in general graphs, randomized in $\mathcal{O}(f * occurences)$ with f time to find one occurence [Meeks '16].

this talk: deterministic worst case output size optimal
for any length in degenerate graphs in $\mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$

Outline of the algorithm: given a k-degenerate graph

- Construct an acyclic orientation of the graph with out-degree bounded by k

Outline of the algorithm: given a k-degenerate graph

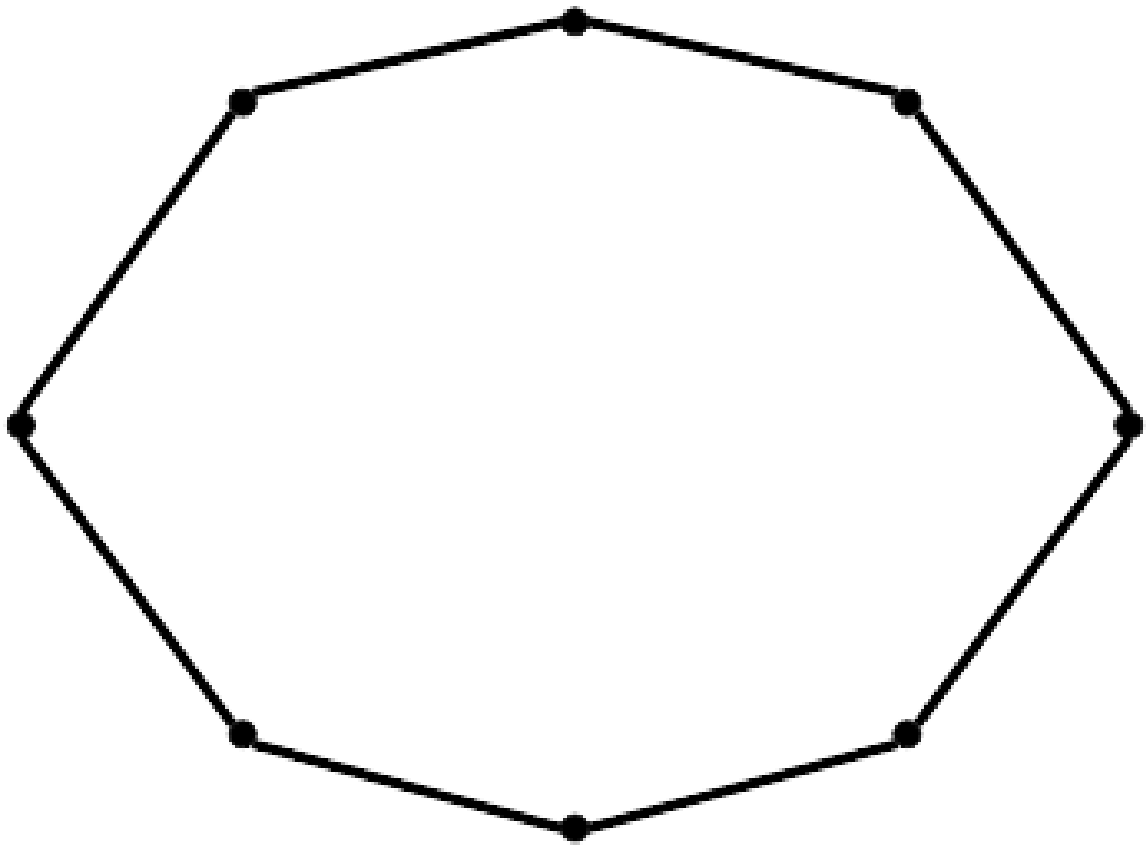
- Construct an acyclic orientation of the graph with out-degree bounded by k
- Prove that any cycle can be decomposed into small easily computable parts

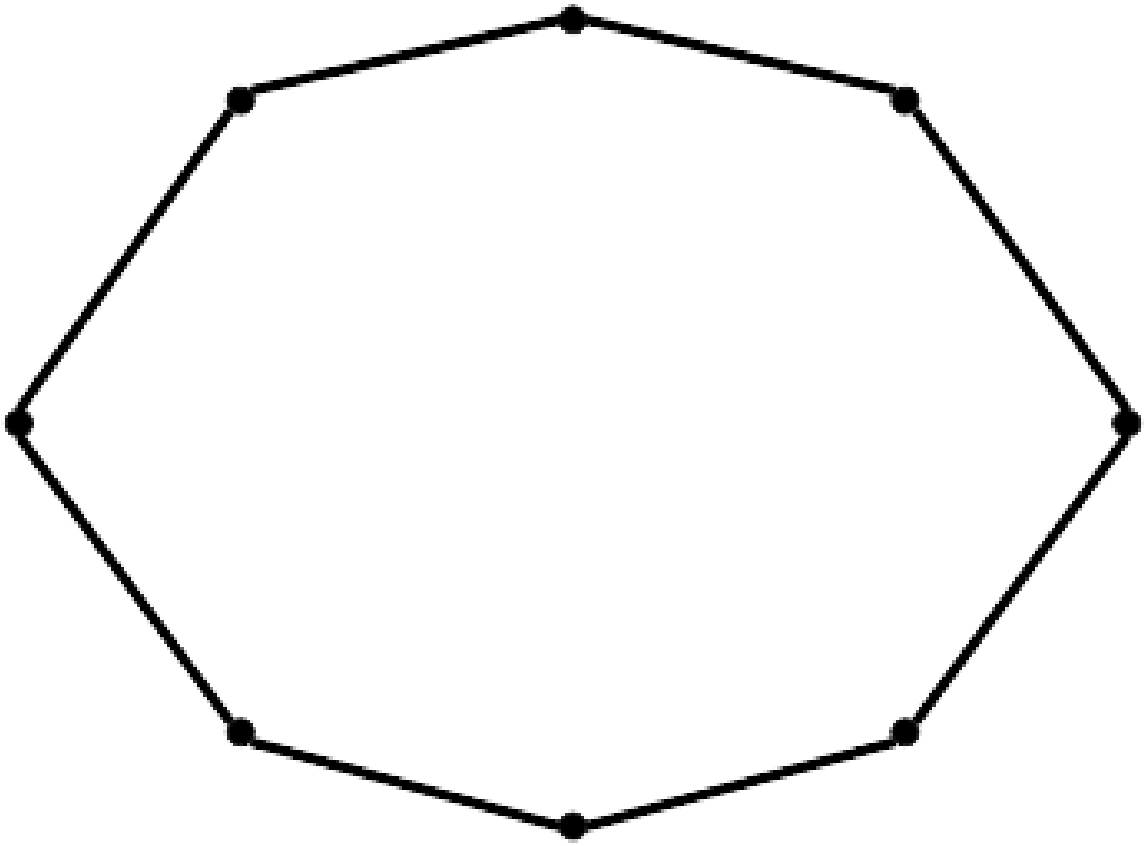
Outline of the algorithm: given a k-degenerate graph

- Construct an acyclic orientation of the graph with out-degree bounded by k
- Prove that any cycle can be decomposed into small easily computable parts
- Form all possible cycles by computing all these possible parts and their possible combinations

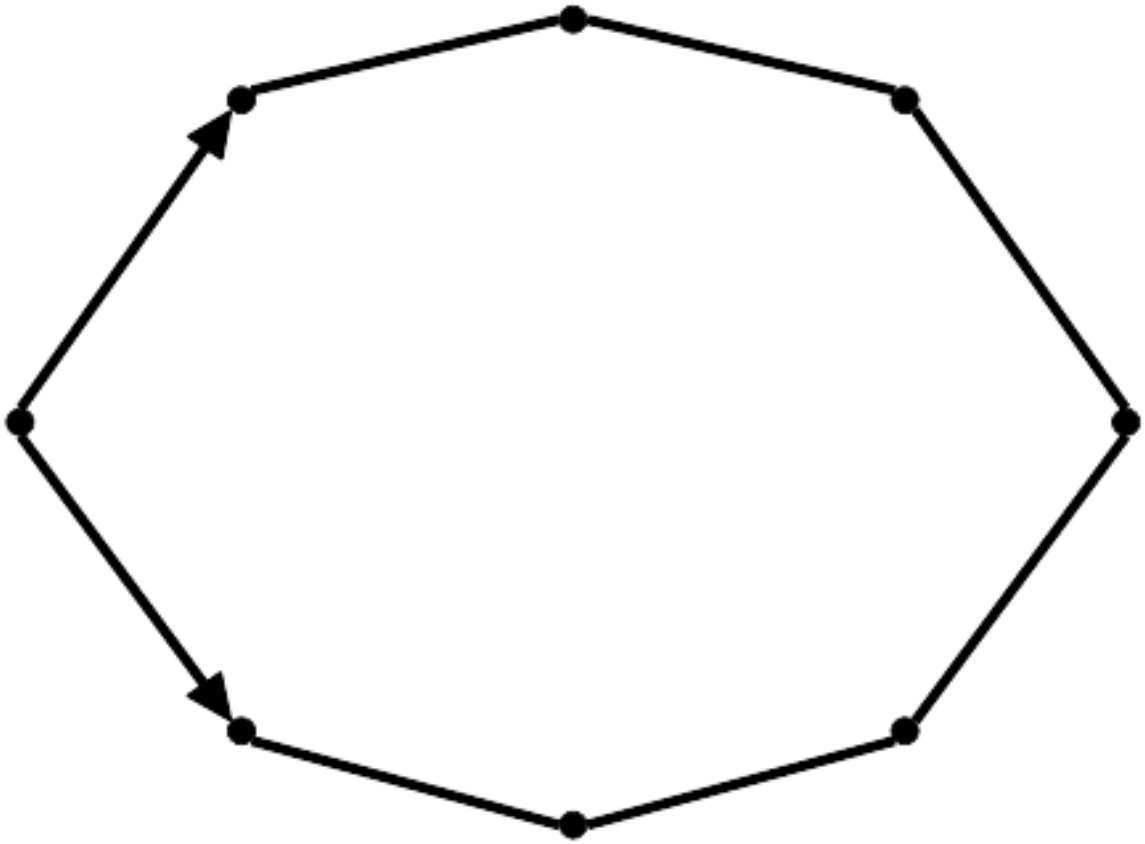
Outline of the algorithm: given a k-degenerate graph

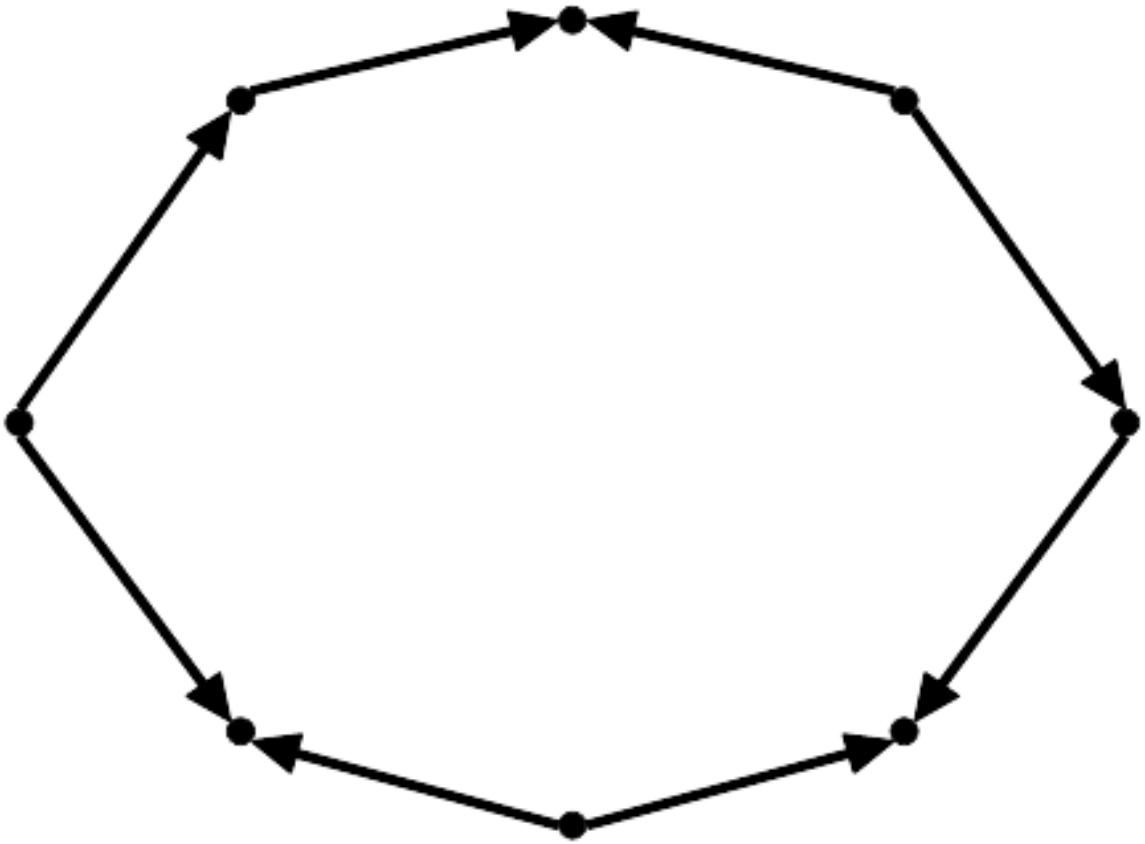
- Construct an acyclic orientation of the graph with out-degree bounded by k
- Prove that any cycle can be decomposed into small easily computable parts
- Form all possible cycles by computing all these possible parts and their possible combinations
- Using properties of the decomposition output each cycle exactly once

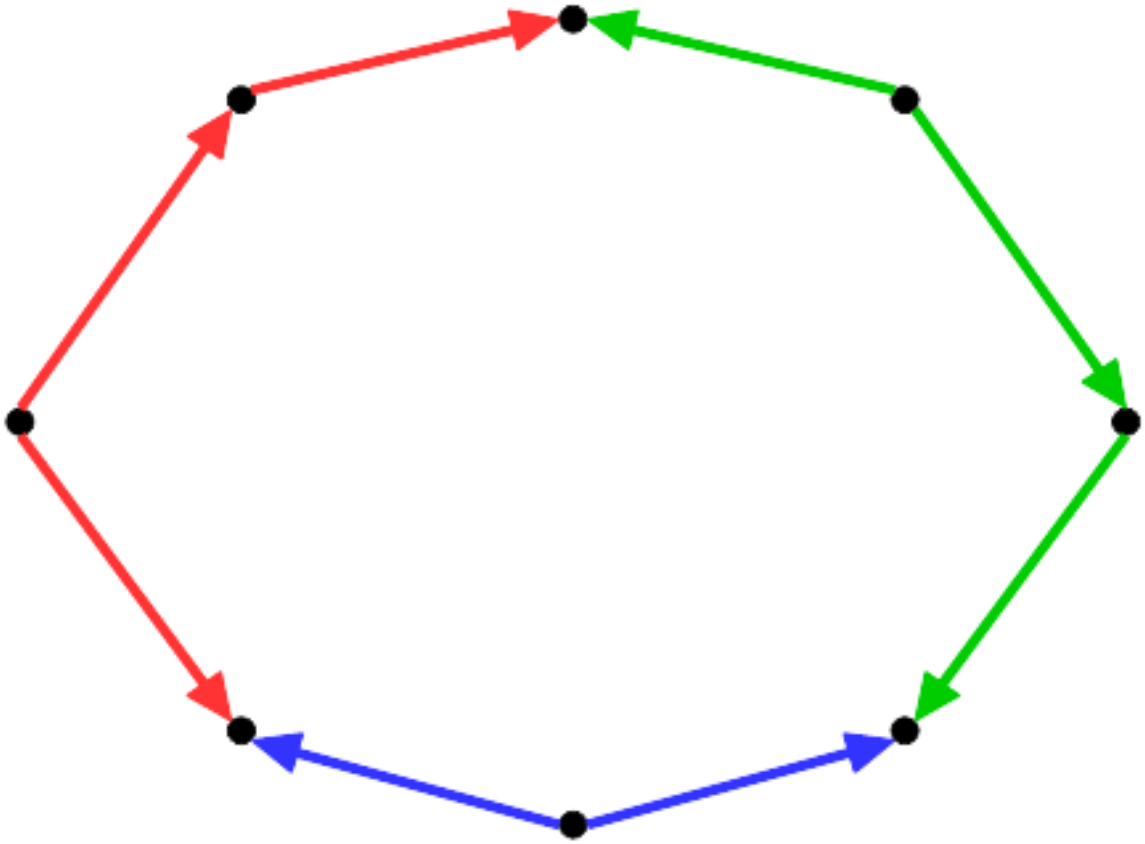




Acyclic orientation: $\exists$ some vertex with two out-going edges

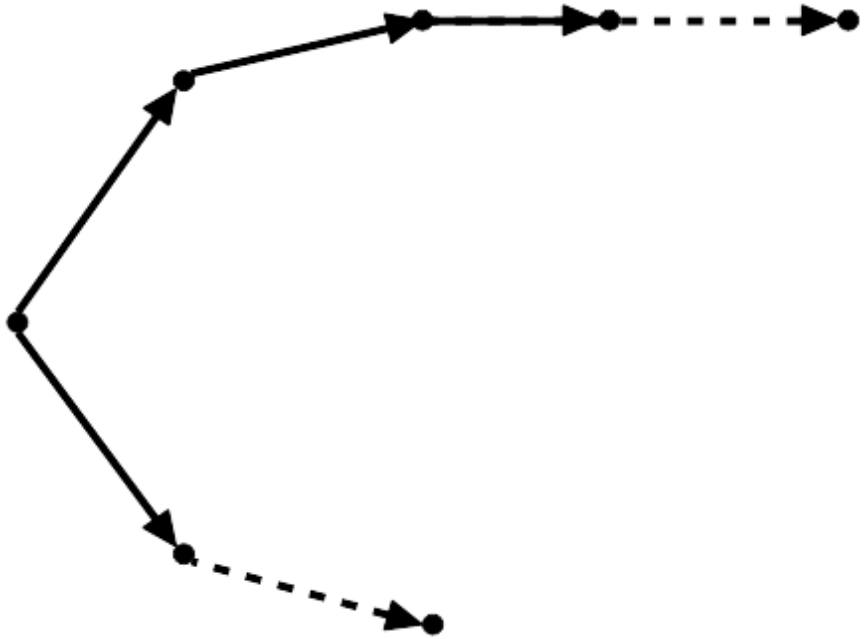






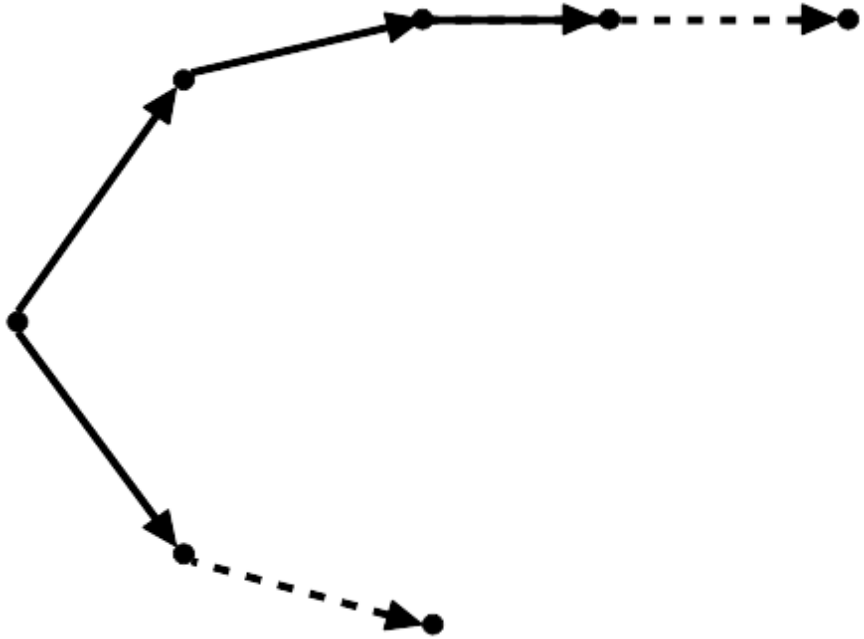
Definition (t-path)

t-path of size (i, j) : two oriented paths of length i and j starting from the same vertex



Definition (t-path)

t-path of size (i, j) : two oriented paths of length i and j starting from the same vertex



Remark

in k-degenerate graphs, at most nk^{i+j} such paths

Algorithm: brute force is optimal

input: graph G k-degenerate

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all possible acyclic orientations of a p-length simple cycle

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all possible acyclic orientations of a p-length simple cycle
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all possible acyclic orientations of a p-length simple cycle
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$
    3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G.

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all possible acyclic orientations of a p-length simple cycle
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$
    3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G.
    4. Construct all possible combinations of these t-paths in time $N(|t_1|) * N(|t_2|) * ... * N(|t_r|)$

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all possible acyclic orientations of a p-length simple cycle
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$
    3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G.
    4. Construct all possible combinations of these t-paths in time $N(|t_1|) * N(|t_2|) * ... * N(|t_r|)$
    5. For each combination: check if it is a cycle + uniqueness

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$
   5. For each combination: check if it is a cycle + uniqueness

Complexity?

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$
   5. For each combination: check if it is a cycle + uniqueness

Complexity?

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
    3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G
    4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$
    5. For each combination: check if it is a cycle + uniqueness

Complexity?

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G $\mathcal{O}(nk^p)$
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$
   5. For each combination: check if it is a cycle + uniqueness

Complexity?

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G $\mathcal{O}(nk^p)$
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$ $\mathcal{O}(n^r k^p)$
   5. For each combination: check if it is a cycle + uniqueness

Complexity?

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G $\mathcal{O}(nk^p)$
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$ $\mathcal{O}(n^r k^p)$
   5. For each combination: check if it is a cycle + uniqueness $\mathcal{O}(p)$

Complexity?

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G $\mathcal{O}(nk^p)$
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$ $\mathcal{O}(n^r k^p)$
   5. For each combination: check if it is a cycle + uniqueness $\mathcal{O}(p)$

total time: $\mathcal{O}(n^r k^p)$

Algorithm: brute force is optimal

input: graph G k-degenerate

1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G $\mathcal{O}(nk^p)$
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$ $\mathcal{O}(n^r k^p)$
   5. For each combination: check if it is a cycle + uniqueness $\mathcal{O}(p)$

total time: $\mathcal{O}(n^r k^p)$

$+ \; r \leq \lfloor p/2 \rfloor \implies \mathcal{O}(n^{\lfloor p/2 \rfloor} k^p)$

Algorithm: brute force is optimal

input: graph G k-degenerate

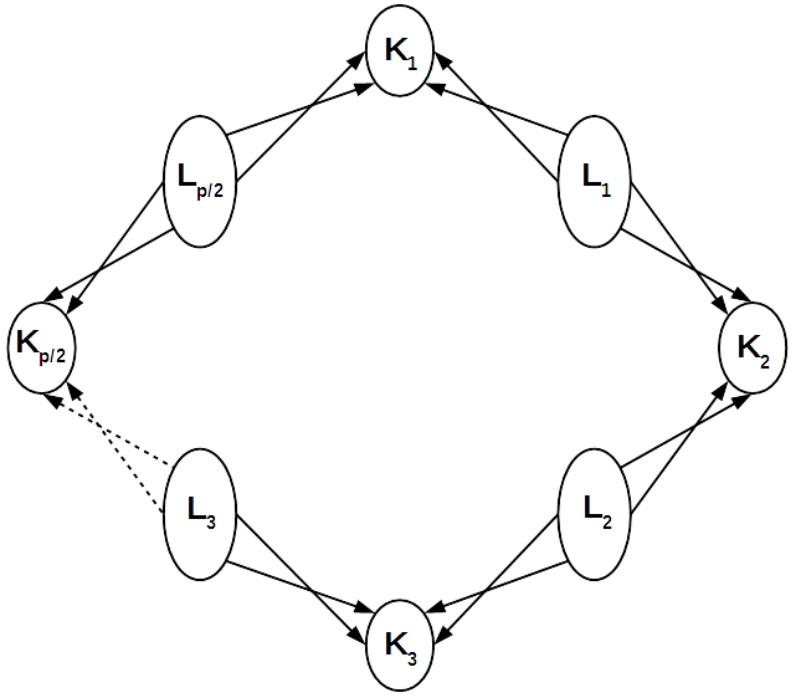
1. Compute all acyclic orientations of a p-length simple cycle $\mathcal{O}(2^p)$
2. For each orientation find a t-path decomposition $t_1, t_2, ..., t_r$ $\mathcal{O}(p)$
   3. Compute all possible t-paths of size $|t_1|, |t_2|, ..., |t_r|$ in G $\mathcal{O}(nk^p)$
   4. Construct all possible combinations of these t-paths in time $N(t_1) * N(t_2) * ... * N(t_r)$ $\mathcal{O}(n^r k^p)$
   5. For each combination: check if it is a cycle + uniqueness $\mathcal{O}(p)$

total time: $\mathcal{O}(n^r k^p)$

$+ \; r \leq \lfloor p/2 \rfloor \implies \mathcal{O}(n^{\lfloor p/2 \rfloor} k^p)$

$+$ fine tuning $\implies \mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$

Proof of optimality: (for k, p even)



Proof of optimality: (for k, p even)

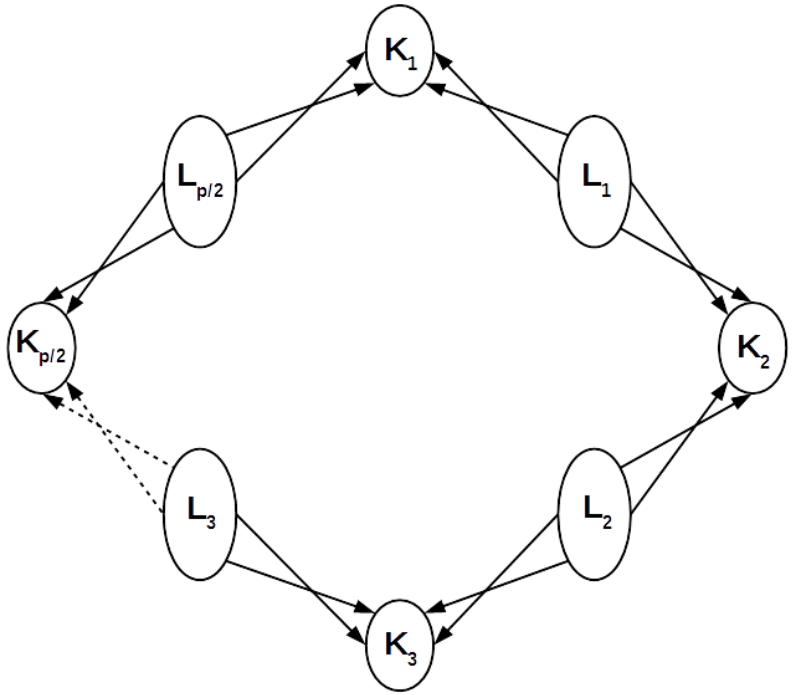


Figure: sets K_i of size k/2, sets L_i of size $h \geq k$

graph is k-degenerate

Proof of optimality: (for k, p even)

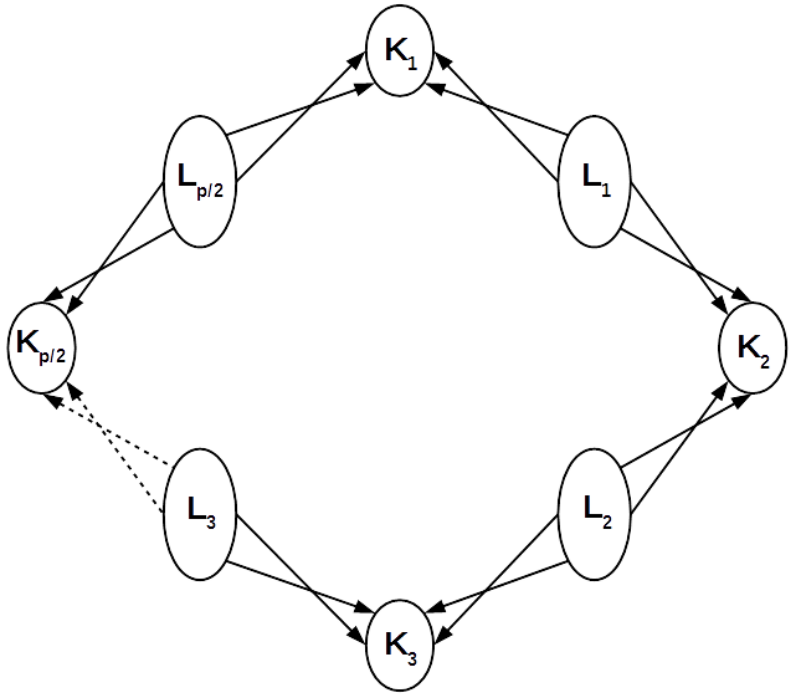


Figure: sets K_i of size k/2, sets L_i of size $h \geq k$

graph is k-degenerate

number of vertices: $n = \frac{p}{2}\frac{k}{2} + h\frac{p}{2} \implies h = \Theta(n)$ if $n \geq kp$

Proof of optimality: (for k, p even)

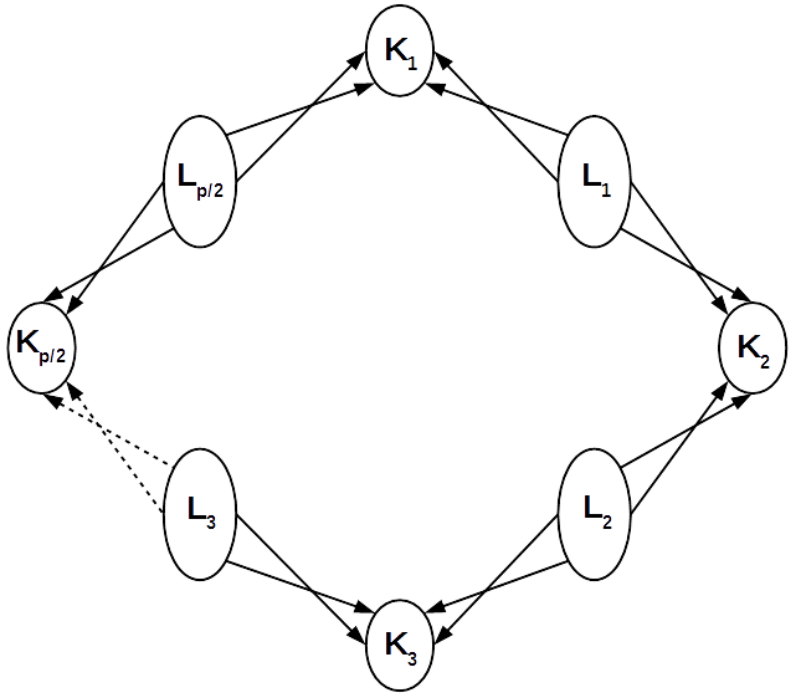


Figure: sets K_i of size k/2, sets L_i of size $h \geq k$

graph is k-degenerate

number of vertices: $n = \frac{p}{2}\frac{k}{2} + h\frac{p}{2} \implies h = \Theta(n)$ if $n \geq kp$

number of cycles: $(\frac{k}{2})^{p/2} h^{p/2} = \Omega(k^{p/2} n^{p/2})$

Open problems:

- List p-cycles in k-degenerate graphs in $\mathcal{O}(occurences)$?
- *enumerating* p-cycles takes time $\mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil})$
  how much faster is *counting*
  - for general graphs:
    $\mathcal{O}(n^{\omega})$ [Alon *et al.* '97] for cycles length ≤ 7
    $\mathcal{O}(n^{\lceil p/2 \rceil + 3})$ [Vassilevska *et al.* '09]
    $\mathcal{O}(n^{0.45p + \mathcal{O}(1)})$ [Bjorklund *et al.* '14]
- what if we do not print cycles ?
  $\mathcal{O}(nk^2)$ for C_4, [Chiba *et al.* '85]
  $\mathcal{O}(nk^3)$ for C_5, [Kowalik '03]
- optimal complexity is attained assuming adjacency matrix, if only adjacency list then $\mathcal{O}(n^{\lfloor p/2 \rfloor} k^{\lceil p/2 \rceil} \log k)$