# IPCO 2014

17th Conference on Integer Programming and Combinatorial Optimization

Location: Bonn, Germany
Date: June 23–25, 2014

www.or.uni-bonn.de/ipco


Submission deadline: November 15, 2013
Program committee chair: Jon Lee
Local organization: Stephan Held, Jens Vygen

Extras:

- summer school (before IPCO)
- welcome reception, Arithmeum
- poster session
- Rhine river cruise with dinner

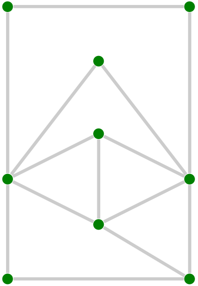
# Smallest two-edge-connected spanning subgraphs and the TSP

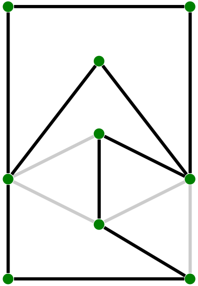
**Jens Vygen**

University of Bonn

(joint work with András Sebő)

August 1, 2013





# Metric TSP

Given a complete graph $G$ and metric weights $c : E(G) \to \mathbb{R}_{\geq 0}$, find a Hamiltonian circuit in $G$ with minimum total weight.

- *NP*-hard
- best known approximation ratio $\frac{3}{2}$ (Christofides [1976])
- no $\frac{123}{122}$-approximation algorithm exists unless $P = NP$ (Karpinski, Lampis, Schmied [2013])
- integrality ratio of subtour relaxation between $\frac{4}{3}$ and $\frac{3}{2}$ (Wolsey [1980]), worst example is instance of Graph-TSP

# Metric TSP

Given a complete graph $G$ and metric weights $c : E(G) \to \mathbb{R}_{\geq 0}$, find a Hamiltonian circuit in $G$ with minimum total weight.

- *NP*-hard
- best known approximation ratio $\frac{3}{2}$ (Christofides [1976])
- no $\frac{123}{122}$-approximation algorithm exists unless $P = NP$ (Karpinski, Lampis, Schmied [2013])
- integrality ratio of subtour relaxation between $\frac{4}{3}$ and $\frac{3}{2}$ (Wolsey [1980]), worst example is instance of Graph-TSP

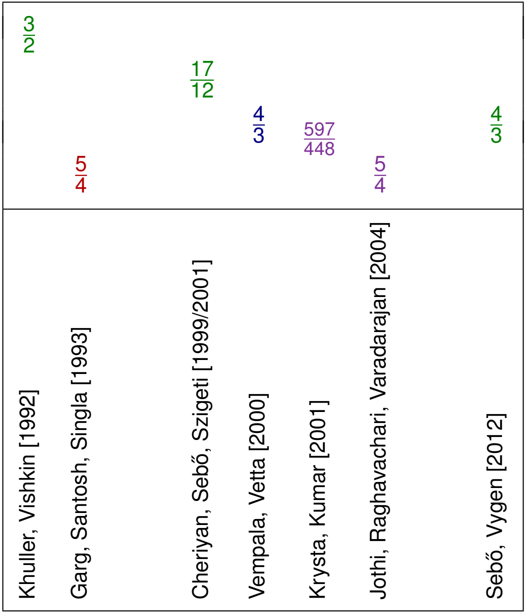
## Graph-TSP (= Eulerian 2ECSS):

- approximation ratio $1.5 - \epsilon$ (Oveis Gharan, Saberi, Singh [2011])
- approximation ratio 1.461 (Mömke, Svensson [2011])
- approximation ratio 1.445 (Mucha [2012])
- approximation ratio 1.4 (Sebő, Vygen [2012])

# The unfortunate history of 2ECSS approximation

| | |
|---|---|
| $\frac{3}{2}$ | Khuller, Vishkin [1992] |
| $\frac{5}{4}$ | Garg, Santosh, Singla [1993] |
| $\frac{17}{12}$ | Cheriyan, Sebő, Szigeti [1999/2001] |
| $\frac{4}{3}$ | Vempala, Vetta [2000] |
| $\frac{597}{448}$ | Krysta, Kumar [2001] |
| $\frac{5}{4}$ | Jothi, Raghavachari, Varadarajan [2004] |
| $\frac{4}{3}$ | Sebő, Vygen [2012] |

# The unfortunate history of 2ECSS approximation



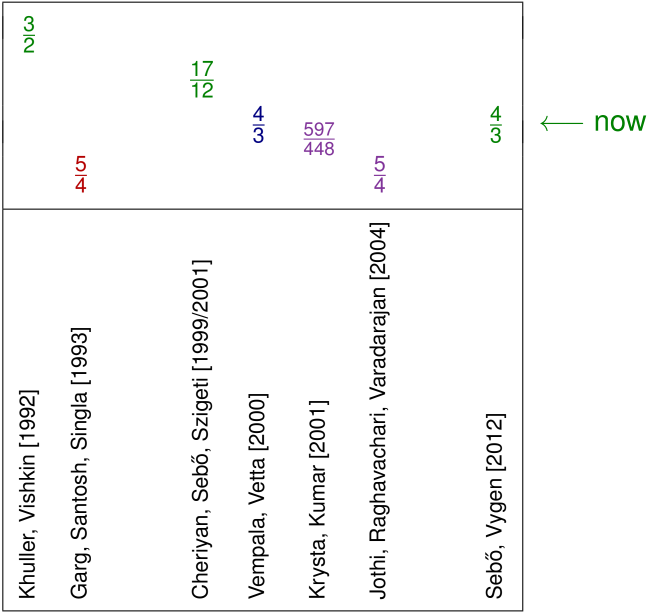

correct proof

wrong proof

incomplete proof

no proof

# The unfortunate history of 2ECSS approximation




correct proof

wrong proof

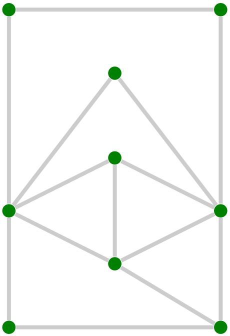
incomplete proof

no proof

# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

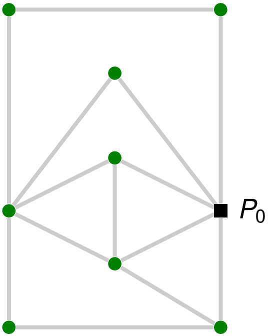
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

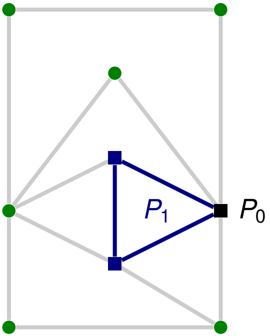
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

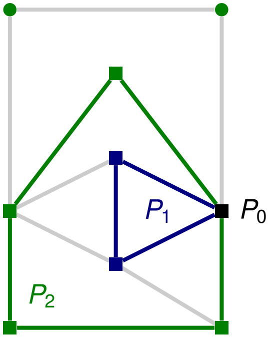
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

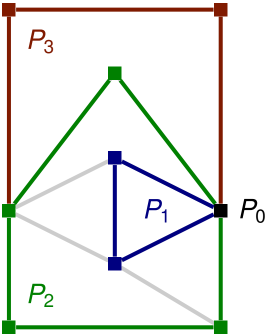
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

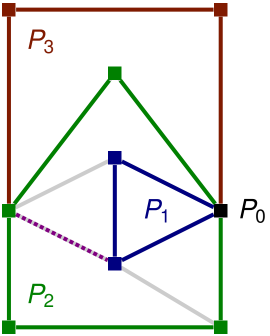
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

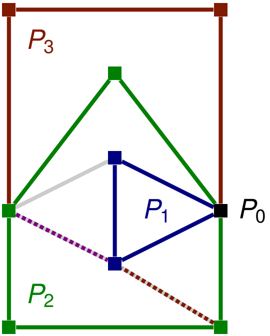
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

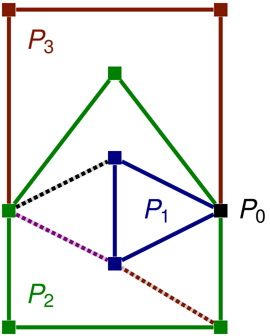
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

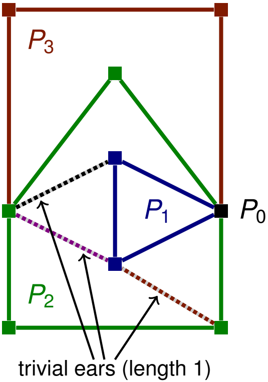
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

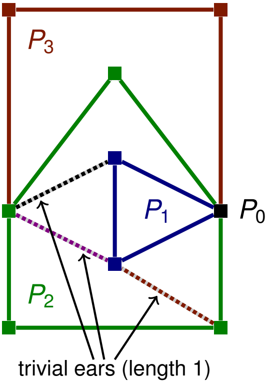
- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.

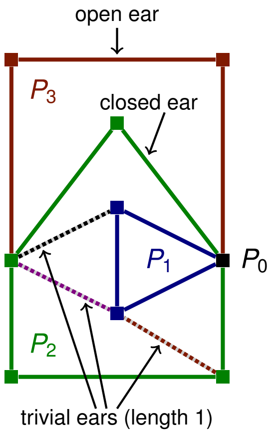



- A graph is 2-edge-connected iff it has an ear-decomposition.

# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.

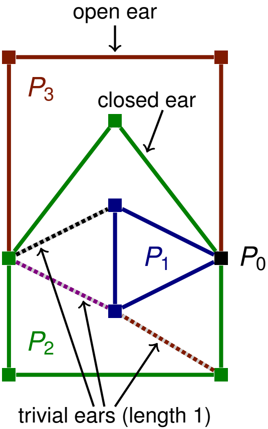



- A graph is 2-edge-connected iff it has an ear-decomposition.

# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.

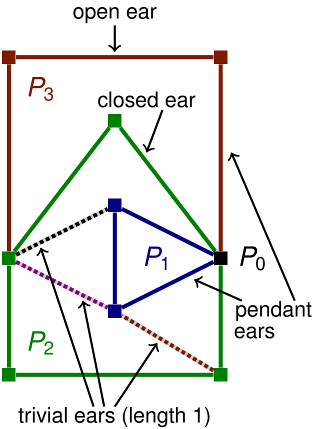



- A graph is 2-edge-connected iff it has an ear-decomposition.
- A graph is 2-vertex-connected iff it has an open ear-decomposition. ($P_2, \ldots, P_k$ are all open ears = paths.)

# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.

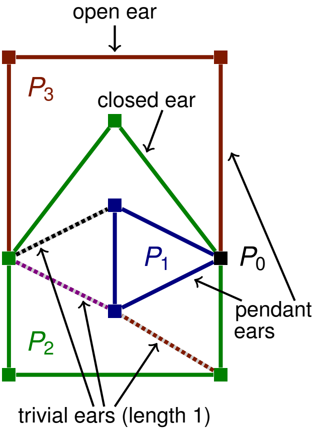



- A graph is 2-edge-connected iff it has an ear-decomposition.
- A graph is 2-vertex-connected iff it has an open ear-decomposition.
- A nontrivial ear is called pendant if none of its internal vertices is endpoint of another nontrivial ear.

# Ear-decompositions

Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.

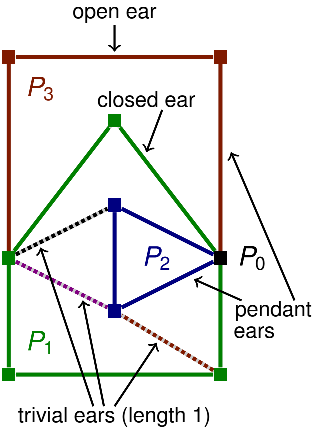



- A graph is 2-edge-connected iff it has an ear-decomposition.
- A graph is 2-vertex-connected iff it has an open ear-decomposition.
- A nontrivial ear is called pendant if none of its internal vertices is endpoint of another nontrivial ear.
- W.l.o.g., pendant ears come last, followed only by trivial ears.

# Ear-decompositions

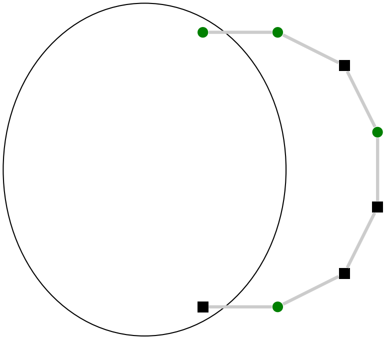
Write $G = P_0 + P_1 + \cdots + P_k$, where $P_0$ is a single vertex, and each $P_i$ ($i = 1, \ldots, k$) is either

- a circuit sharing exactly one vertex with $P_0 + \cdots + P_{i-1}$, or
- a path sharing exactly its endpoints with $P_0 + \cdots + P_{i-1}$.



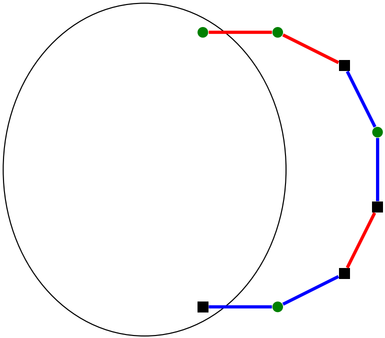

- A graph is 2-edge-connected iff it has an ear-decomposition.
- A graph is 2-vertex-connected iff it has an open ear-decomposition.
- A nontrivial ear is called pendant if none of its internal vertices is endpoint of another nontrivial ear.
- W.l.o.g., pendant ears come last, followed only by trivial ears.

# Ear-decompositions for $T$-joins



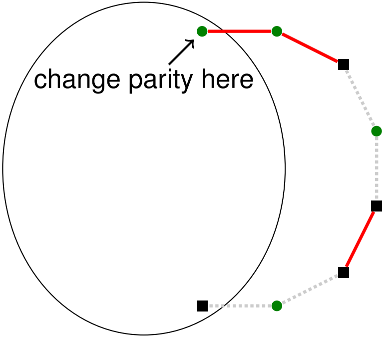
- Ear induction:

# Ear-decompositions for $T$-joins



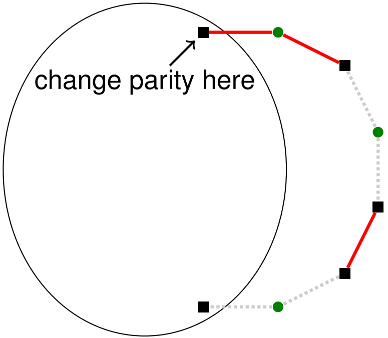
- Ear induction:
- Split pendant ear at the vertices that have wrong parity so far

# Ear-decompositions for $T$-joins




- Ear induction:
- Split pendant ear at the vertices that have wrong parity so far
- Take smaller part

# Ear-decompositions for $T$-joins



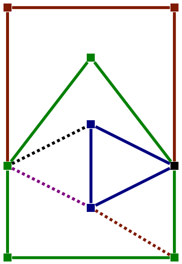

- Ear induction:
- Split pendant ear at the vertices that have wrong parity so far
- Take smaller part

This yields a $T$-join with at most $\frac{1}{2}(n - 1 + k_{\mathrm{even}})$ edges, where $n = |V(G)|$ and $k_{\mathrm{even}}$ is the number of even ears.

# Ear-decompositions for 2ECSS

Simple algorithm for 2ECSS:

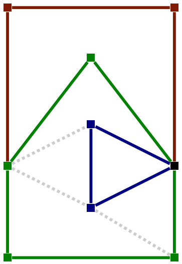
- compute an ear-decomposition
- delete all trivial ears.



# Ear-decompositions for 2ECSS

Simple algorithm for 2ECSS:

- compute an ear-decomposition
- delete all trivial ears.



The remaining number of edges is at most

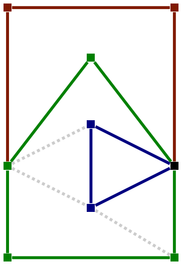
$$\frac{5}{4}(n-1)+\frac{3}{4}k_2+\frac{1}{2}k_3+\frac{1}{4}k_4,$$

where $n=|V(G)|$ and $k_i$ is the number of ears of length i.

# Ear-decompositions for 2ECSS

Simple algorithm for 2ECSS:

- compute an ear-decomposition
- delete all trivial ears.



The remaining number of edges is at most

$$\frac{5}{4}(n-1) + \frac{3}{4}k_2 + \frac{1}{2}k_3 + \frac{1}{4}k_4,$$

where $n = |V(G)|$ and $k_i$ is the number of ears of length i.

So:

- even ears are bad, and
- 3-ears are bad.

# Ear-decompositions with fewest even ears

For a 2-edge-connected graph $G$, let $\varphi(G)$ denote the minimum number of even ears in an ear-decomposition of $G$.

**Theorem (Frank [1993])**

*Let $G$ be a 2-edge-connected graph. Then an ear-decomposition with $\varphi(G)$ even ears can be computed in polynomial time,*

# Ear-decompositions with fewest even ears

For a 2-edge-connected graph $G$, let $\varphi(G)$ denote the minimum number of even ears in an ear-decomposition of $G$.

### Theorem (Frank [1993])

*Let $G$ be a 2-edge-connected graph. Then an ear-decomposition with $\varphi(G)$ even ears can be computed in polynomial time, and*

$$\tfrac{|V(G)|-1+\varphi(G)}{2} = \max\Big\{\min\{|J| : J \text{ is a } T\text{-join}\} : T \subseteq V(G),\ |T| \text{ even}\Big\}.$$

Note:

- Every 2ECSS contains at least $\varphi(G)$ even (thus: nontrivial) ears.
- So every 2ECSS contains at least $n - 1 + \varphi(G)$ edges.

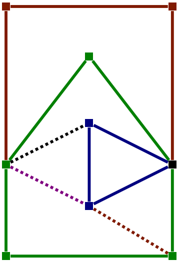
# Ear-decompositions for 2ECSS

Simple algorithm for 2ECSS:

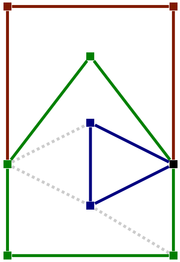
- compute an ear-decomposition
- delete all trivial ears.

The remaining number of edges is at most

$$\frac{5}{4}(n-1) + \frac{3}{4}k_2 + \frac{1}{2}k_3 + \frac{1}{4}k_4$$



# Ear-decompositions for 2ECSS



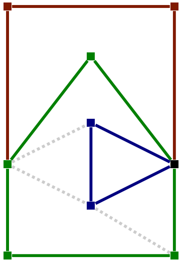
Simple algorithm for 2ECSS:

- compute an ear-decomposition
- delete all trivial ears.

The remaining number of edges is at most

$$
\begin{aligned}
& \tfrac{5}{4}(n-1) + \tfrac{3}{4}k_2 + \tfrac{1}{2}k_3 + \tfrac{1}{4}k_4 \\
\leq \quad & \tfrac{5}{4}(n-1+k_{\text{even}}) + \tfrac{1}{2}k_3 \\
= \quad & \tfrac{5}{4}(n-1+\varphi(G)) + \tfrac{1}{2}k_3
\end{aligned}
$$

# Ear-decompositions for 2ECSS



Simple algorithm for 2ECSS:

- compute an ear-decomposition
- delete all trivial ears.

The remaining number of edges is at most

$$\begin{aligned} & \tfrac{5}{4}(n-1) + \tfrac{3}{4}k_2 + \tfrac{1}{2}k_3 + \tfrac{1}{4}k_4 \\ \leq \quad & \tfrac{5}{4}(n-1+k_{\text{even}}) + \tfrac{1}{2}k_3 \\ = \quad & \tfrac{5}{4}(n-1+\varphi(G)) + \tfrac{1}{2}k_3 \end{aligned}$$

Henceforth (for this talk only) assume $\varphi(G) = 0$.
In other words, $G$ is factor-critical (Lovász [1972]).

Note: 3-ears are still bad.

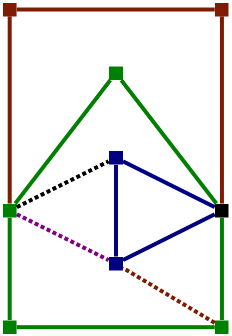
# Nice ear-decompositions

An ear-decomposition is called nice if

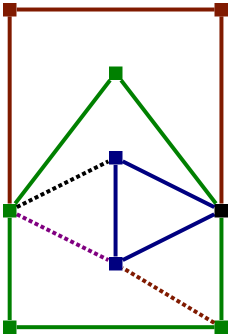
(i) the number of even ears is minimum,

(ii) all short ears (length 2 or 3) are pendant,

(iii) and there are no edges connecting internal vertices of different short ears.



# Nice ear-decompositions



An ear-decomposition is called nice if

(i) the number of even ears is minimum,

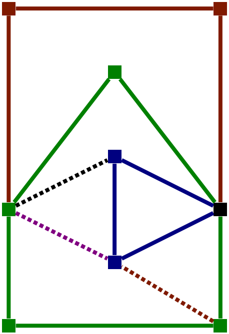
(ii) all short ears (length 2 or 3) are pendant,

(iii) and there are no edges connecting internal vertices of different short ears.

Lemma (Cheriyan, Sebő, Szigeti [2001])

*A nice ear-decomposition can be computed in polynomial time.*

# Nice ear-decompositions



An ear-decomposition is called nice if

(i) the number of even ears is minimum,

(ii) all short ears (length 2 or 3) are pendant,

(iii) and there are no edges connecting internal vertices of different short ears.

Lemma (Cheriyan, Sebő, Szigeti [2001])

*A nice ear-decomposition can be computed in polynomial time.*

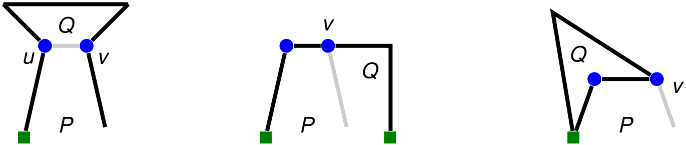
Sketch of Proof (for $\varphi(G) = 0$):

- Compute an open odd ear-decomp. (Lovász, Plummer [1986])
- Replace non-pendant short ears
- Replace adjacent short ears

$\square$

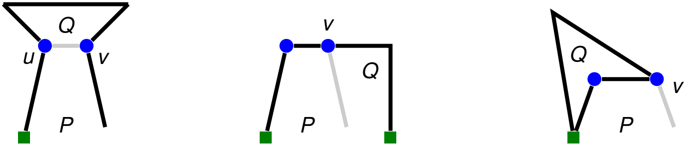
# Sketch of proof (some details)

- Replace non-pendant short ears



# Sketch of proof (some details)

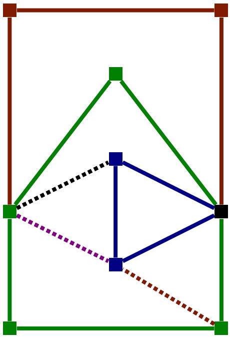
- Replace non-pendant short ears




- Replace adjacent short ears

# Optimizing short ears

- Adding all short ears leaves some number of connected components

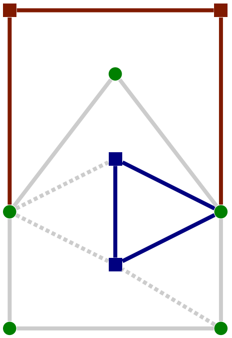


Recall: An ear-decomposition is called nice if

(i) the number of even ears is minimum,
(ii) all short ears (length 2 or 3) are pendant,
(iii) and there are no edges connecting internal vertices of different short ears.

# Optimizing short ears

- Adding all short ears leaves some number of connected components

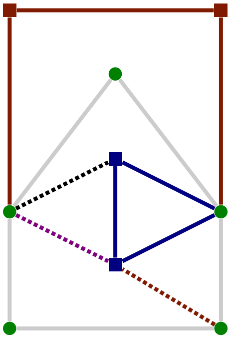


Recall: An ear-decomposition is called nice if

(i) the number of even ears is minimum,
(ii) all short ears (length 2 or 3) are pendant,
(iii) and there are no edges connecting internal vertices of different short ears.

# Optimizing short ears

- Adding all short ears leaves some number of connected components
- Internal vertices of short ears may be incident to trivial ears

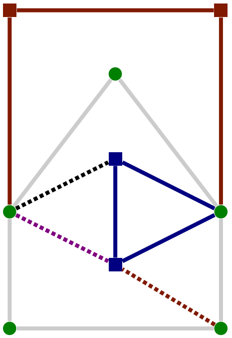


Recall: An ear-decomposition is called nice if

(i) the number of even ears is minimum,
(ii) all short ears (length 2 or 3) are pendant,
(iii) and there are no edges connecting internal vertices of different short ears.

# Optimizing short ears

- Adding all short ears leaves some number of connected components
- Internal vertices of short ears may be incident to trivial ears
- These can be used to replace some short ears by other short ears

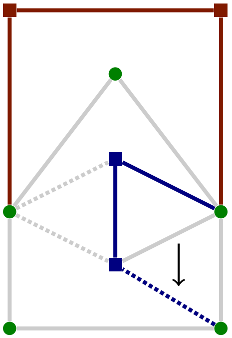


**Recall:** An ear-decomposition is called nice if

(i) the number of even ears is minimum,
(ii) all short ears (length 2 or 3) are pendant,
(iii) and there are no edges connecting internal vertices of different short ears.

# Optimizing short ears

- Adding all short ears leaves some number of connected components
- Internal vertices of short ears may be incident to trivial ears
- These can be used to replace some short ears by other short ears
- Goal: minimize the resulting number of connected components

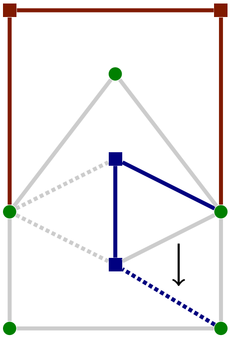


Recall: An ear-decomposition is called nice if

(i) the number of even ears is minimum,
(ii) all short ears (length 2 or 3) are pendant,
(iii) and there are no edges connecting internal vertices of different short ears.

# Optimizing short ears

- Adding all short ears leaves some number of connected components
- Internal vertices of short ears may be incident to trivial ears
- These can be used to replace some short ears by other short ears
- Goal: minimize the resulting number of connected components

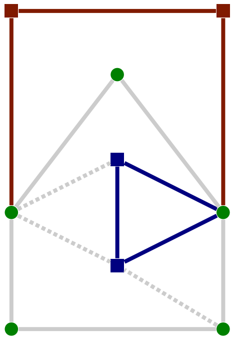


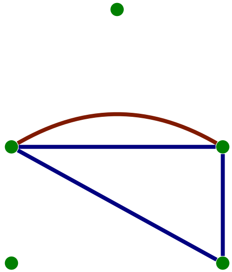
Note: Replacing some short ears by other ears (with the same internal vertices) will maintain a nice ear-decomposition.

Recall: An ear-decomposition is called nice if

(i) the number of even ears is minimum,
(ii) all short ears (length 2 or 3) are pendant,
(iii) and there are no edges connecting internal vertices of different short ears.

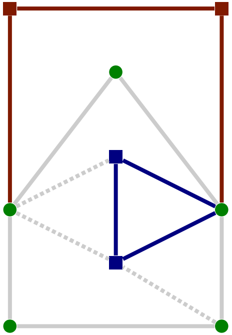
# First solution: matroid intersection

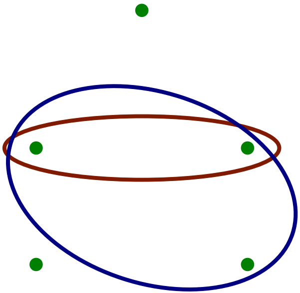




- For each pendant ear (= color), represent each possible variant by an edge connecting its two endpoints
- Pick an edge for each color, so that the edges form a forest
- Intersection of partition matroid and graphic matroid (Rado [1942], Edmonds [1970])

# Second solution: forest representative systems





- For each pendant ear (= color), consider the set of endpoints of the variants. In this hypergraph:
- Find a forest representative system (Lovász [1970])
- This leads to useful ears
- We have an algorithm with runtime $O(|V(G)||E(G)|)$

# New algorithm for 2ECSS

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.

Note: number of even ears is minimum, all short ears are pendant

- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

# New algorithm for 2ECSS

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.

Note: number of even ears is minimum, all short ears are pendant

- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

**Theorem**

*The new algorithm yields a tour with at most $\frac{3}{2}L - \pi$ edges, where $L$ is a lower bound on the number of edges in any 2ECSS, and $\pi$ is the number of pendant ears (after optimization).*

# New algorithm for 2ECSS

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.

Note: number of even ears is minimum, all short ears are pendant

$$\left.\begin{array}{l}\blacktriangleright\ \text{Take all edges of pendant ears.}\\ \blacktriangleright\ \text{Add edges to obtain connectivity.}\end{array}\right\} L + \pi_{\text{long}}$$

- Add edges to correct parity.

### Theorem

*The new algorithm yields a tour with at most $\frac{3}{2}L - \pi$ edges, where L is a lower bound on the number of edges in any 2ECSS, and $\pi$ is the number of pendant ears (after optimization).*

# New algorithm for 2ECSS

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.

Note: number of even ears is minimum, all short ears are pendant

- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

$\Big\}\, L + \pi_{\text{long}}$

$\}\, \frac{1}{2}(n - 1 - 2\pi_{\text{short}} - 4\pi_{\text{long}})$

**Theorem**

*The new algorithm yields a tour with at most $\frac{3}{2}L - \pi$ edges, where L is a lower bound on the number of edges in any 2ECSS, and $\pi$ is the number of pendant ears (after optimization).* $\square$

# New algorithm for 2ECSS

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.

Note: number of even ears is minimum, all short ears are pendant

- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

Alternatively:

- Take all edges of nontrivial ears.

**Theorem**

*The new algorithm yields a tour with at most $\frac{3}{2}L - \pi$ edges, where $L$ is a lower bound on the number of edges in any 2ECSS, and $\pi$ is the number of pendant ears (after optimization).* $\square$

Alternative yields an 2ECSS with at most $\frac{5}{4}L + \frac{1}{2}\pi$ edges.

$\longrightarrow$ The better of the two 2ECSSs has at most $\frac{4}{3}L$ edges.

# New algorithm for TSP

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.

- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

# New algorithm for TSP

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.
- Delete all 1-ears. In each of the resulting blocks:
- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

# New algorithm for TSP

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.
- Delete all 1-ears. In each of the resulting blocks:
- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

### Theorem

*In each block, this algorithm yields a tour with at most $\frac{3}{2}L - \pi$ edges, where L is a lower bound on the number of edges in any 2ECSS, and $\pi$ is the number of pendant ears (after optimization).*

# New algorithm for TSP

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.
- Delete all 1-ears. In each of the resulting blocks:
- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

Alternatively:

- Apply lemma of Mömke-Svensson.

### Theorem

*In each block, this algorithm yields a tour with at most* $\frac{3}{2}L - \pi$ *edges, where* $L$ *is a lower bound on the number of edges in any 2ECSS, and* $\pi$ *is the number of pendant ears (after optimization).*

# New algorithm for TSP

- Compute a nice ear-decomposition.
- Optimize short ears so that they serve best for connectivity.
- Delete all 1-ears. In each of the resulting blocks:
- Take all edges of pendant ears.
- Add edges to obtain connectivity.
- Add edges to correct parity.

Alternatively:

- Apply lemma of Mömke-Svensson.

**Theorem**

*In each block, this algorithm yields a tour with at most $\frac{3}{2}L - \pi$ edges, where $L$ is a lower bound on the number of edges in any 2ECSS, and $\pi$ is the number of pendant ears (after optimization).*

**Theorem**

*Mömke-Svensson yields a tour with at most $\frac{4}{3}L + \frac{2}{3}\pi$ edges.*

$\longrightarrow$ The better of the two tours has at most $\frac{7}{5}L$ edges.

# Open problems

## 2ECSS

- improve approximation ratio
  (combining with ideas from Vempala, Vetta [2000]?)
- improve on 2-approximation for weighted 2ECSS
  (due to Khuller, Vishkin [1994])
- determine integrality ratio of the natural LP relaxation

## TSP

- improve approximation ratio, determine integrality ratio
- extend to general metric TSP (beat Christofides [1976])
- extend to directed graphs (constant factor?)

## $T$-tours $\supseteq$ $s$-$t$-path-TSP

- find $\frac{3}{2}$-approximation algorithm for the weighted case

# Open problems

## 2ECSS

- improve approximation ratio
  (combining with ideas from Vempala, Vetta [2000]?)
- improve on 2-approximation for weighted 2ECSS
  (due to Khuller, Vishkin [1994])
- determine integrality ratio of the natural LP relaxation

## TSP

- improve approximation ratio, determine integrality ratio
- extend to general metric TSP (beat Christofides [1976])
- extend to directed graphs (constant factor?)

## $T$-tours $\supseteq$ $s$-$t$-path-TSP

- find $\frac{3}{2}$-approximation algorithm for the weighted case

# Thank you!

# Open problems

## 2ECSS

- improve approximation ratio
  (combining with ideas from Vempala, Vetta [2000]?)
- improve on 2-approximation for weighted 2ECSS
  (due to Khuller, Vishkin [1994])
- determine integrality ratio of the natural LP relaxation

## TSP

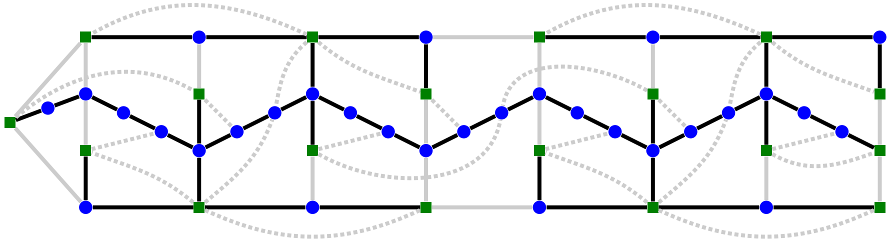
- improve approximation ratio, determine integrality ratio
- extend to general metric TSP (beat Christofides [1976])
- extend to directed graphs (constant factor?)

## $T$-tours $\supseteq$ $s$-$t$-path-TSP

- find $\frac{3}{2}$-approximation algorithm for the weighted case

# Thank you!

# Tight example for 2ECSS



$L = n = OPT = 24k$ (Here $k = 2$.)
$\varphi(G) = 1$
$\pi = 4k = \frac{1}{6}L$.

Algorithm computes solution with $32k - 1$ edges.