Approximating Multiple Edge Insertion and the Crossing Number

Petr Hliněný

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Rep.

joint work with **Markus Chimani**
Osnabrück University, Germany

0 Bit of History for Start

A WW II story

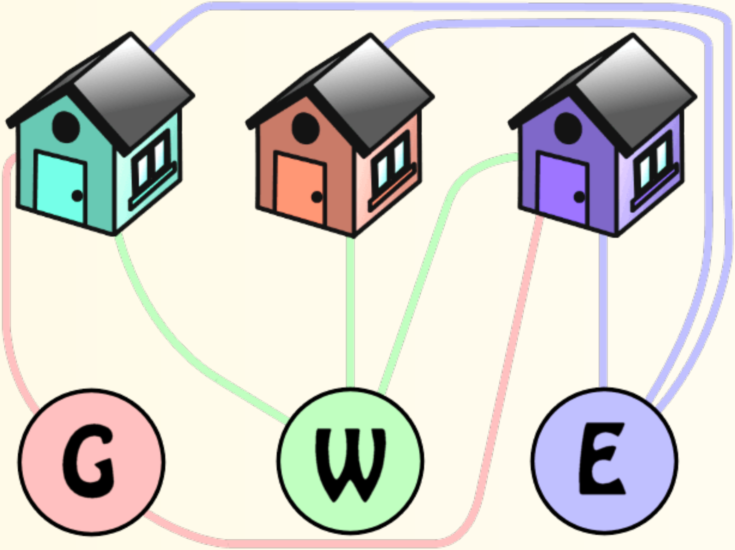
“*There were some kilns where the bricks were made and some open storage yards where the bricks were stored. All the kilns were connected by rail with all the storage yards. The bricks were carried on small wheeled trucks to the storage yards... the work was not difficult; the trouble was only at the crossings. The trucks generally jumped the rails there, and the bricks fell out of them; in short this caused a lot of trouble and loss of time... the idea occurred to me that this loss of time could have been minimized if the number of crossings of the rails had been minimized.*

But what is the minimum number of crossings?

... This problem has become a notoriously difficult unsolved problem.”

Pál Turán, *A note of welcome.*
Journal of Graph Theory (1977)

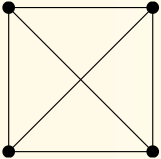
Or, can you avoid all the crossings?

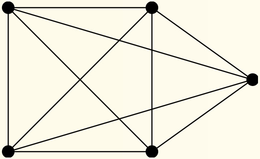


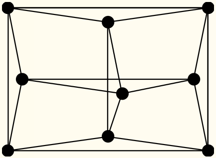
1 Graph Crossing Number

Definition. *Drawing of a graph G*:

- The vertices of G are distinct points,
  and every edge $e = uv \in E(G)$ is a simple curve joining u to v.
- No edge passes through another vertex,
  and no three edges intersect in a common point.



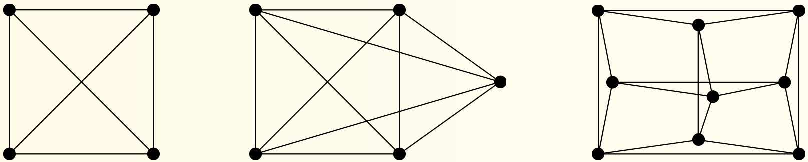




1 Graph Crossing Number

Definition. *Drawing of a graph G*:

- The vertices of G are distinct points,
  and every edge $e = uv \in E(G)$ is a simple curve joining u to v.
- No edge passes through another vertex,
  and no three edges intersect in a common point.

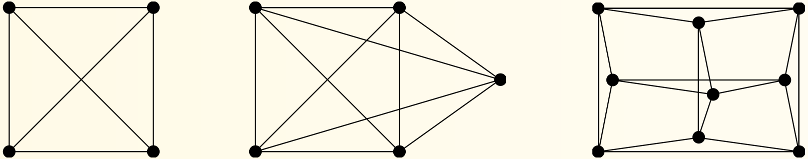


Definition. **Crossing number** $cr(G)$
is the smallest number of edge crossings in a drawing of G.

1 Graph Crossing Number

Definition. *Drawing of a graph G*:

- The vertices of G are distinct points, and every edge $e = uv \in E(G)$ is a simple curve joining u to v.
- No edge passes through another vertex, and no three edges intersect in a common point.



Definition. **Crossing number** *cr*(G) is the smallest number of edge crossings in a drawing of G.

Warning. There are slight variations of the definition of crossing number, some giving different numbers! Such as counting *odd-crossing pairs* of edges. [Pelsmajer, Schaeffer, Štefankovič, 2005]...

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]
- The degree-3 and *minor-monotone* cases; [**PH**, 2004]

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]
- The degree-3 and *minor-monotone* cases; [**PH**, 2004]
- Even fixed *rotation scheme*; [**Pelsmajer, Schaeffer, Štefankovič**, 2007]

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]
- The degree-3 and *minor-monotone* cases; [**PH**, 2004]
- Even fixed *rotation scheme*; [**Pelsmajer, Schaeffer, Štefankovič**, 2007]

Fixed parameter tractability

- *FPT* when parameterized by itself (but totally impractical);
  [**Grohe**, 2001], [**Kawarabayashi and Reed**, 2007]

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]
- The degree-3 and *minor-monotone* cases; [**PH**, 2004]
- Even fixed *rotation scheme*; [**Pelsmajer, Schaeffer, Štefankovič**, 2007]

Fixed parameter tractability

- *FPT* when parameterized by itself (but totally impractical); [**Grohe**, 2001], [**Kawarabayashi and Reed**, 2007]
- However, NO rich *natural graph class / parameter* with nontrivial and yet efficiently computable exact crossing number problem is known. . .

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]
- The degree-3 and *minor-monotone* cases; [**PH**, 2004]
- Even fixed *rotation scheme*; [**Pelsmajer, Schaeffer, Štefankovič**, 2007]

Fixed parameter tractability

- *FPT* when parameterized by itself (but totally impractical);
  [**Grohe**, 2001], [**Kawarabayashi and Reed**, 2007]
- However, NO rich *natural graph class / parameter* with nontrivial and yet efficiently computable exact crossing number problem is known. . .

Approximations, at least?

- Up to factor $\log^3 |V(G)|$ $(\log^2 \cdot)$ for $cr(G)+|V(G)|$ with bounded degs.;
  [**Even, Guha and Schieber**, 2002]

2 How to Compute the Crossing Number

Not easily. . . !

NP-hardness

- The general case (no surprise?); [**Garey and Johnson**, 1983]
- The degree-3 and *minor-monotone* cases; [**PH**, 2004]
- Even fixed *rotation scheme*; [**Pelsmajer, Schaeffer, Štefankovič**, 2007]

Fixed parameter tractability

- *FPT* when parameterized by itself (but totally impractical);
  [**Grohe**, 2001], [**Kawarabayashi and Reed**, 2007]
- However, NO rich *natural graph class / parameter* with nontrivial and yet efficiently computable exact crossing number problem is known. . .

Approximations, at least?

- Up to factor $\log^3 |V(G)|$ ($\log^2 \cdot$) for $cr(G) + |V(G)|$ with bounded degs.;
  [**Even, Guha and Schieber**, 2002]
- No constant factor $c > 1$ -approximation; [**Cabello**, 2013]

Approximating "Small" Crossing Numbers

The case $cr(G) \in o(|V(G)|)$ seems very hard to approximate in general...

– no good general algorithms are known, and

Approximating "Small" Crossing Numbers

The case $cr(G) \in o(|V(G)|)$ seems very hard to approximate in general...

- no good general algorithms are known, and
- no good quantitative lower bounds exist.

Approximating "Small" Crossing Numbers

The case $cr(G) \in o(|V(G)|)$ seems very hard to approximate in general...

– no good general algorithms are known, and
– no good quantitative lower bounds exist.

Close to planarity?

- $cr(G) \in o(|V(G)|)$ means "most of" G is planar... **BUT**

Approximating "Small" Crossing Numbers

The case $cr(G) \in o(|V(G)|)$ seems very hard to approximate in general...

– no good general algorithms are known, and
– no good quantitative lower bounds exist.

Close to planarity?

- $cr(G) \in o(|V(G)|)$ means "most of" G is planar... **BUT**
- Crossing number NP-hard already for *planar graphs plus one edge*!
[**Cabello and Mohar**, 2010]

Approximating "Small" Crossing Numbers

The case $cr(G) \in o(|V(G)|)$ seems very hard to approximate in general...

- no good general algorithms are known, and
- no good quantitative lower bounds exist.

Close to planarity?

- $cr(G) \in o(|V(G)|)$ means "most of" G is planar... **BUT**
- Crossing number NP-hard already for *planar graphs plus one edge*!
  [**Cabello and Mohar**, 2010]

Still, approximations do exist

- Factor $\Delta(G)$ for planar graphs plus one edge; [**PH and Salazar**, 2006]

Approximating "Small" Crossing Numbers

The case $cr(G) \in o(|V(G)|)$ seems very hard to approximate in general...

– no good general algorithms are known, and
– no good quantitative lower bounds exist.

Close to planarity?

- $cr(G) \in o(|V(G)|)$ means "most of" G is planar... **BUT**
- Crossing number NP-hard already for *planar graphs plus one edge*!
  [**Cabello and Mohar**, 2010]

Still, approximations do exist

- Factor $\Delta(G)$ for planar graphs plus one edge; [**PH and Salazar**, 2006]
- Constant-factor for surface-embedded bounded-degree graphs;
  [**Gitler et al**, 2007], [**PH and Salazar**, 2007], [**PH and Chimani**, 2010]

3 Planar Insertion Problems

Keeping "most of" G planar...

Definition. Given a planar graph G and a set F of additional edges (vert.). Find a *drawing of* $G + F$ minimizing the edge crossings *ins*$(\boldsymbol{G}, \boldsymbol{E})$ such that the subdrawing of G is plane.

3 Planar Insertion Problems

Keeping "most of" G planar...

Definition. Given a planar graph G and a set F of additional edges (vert.). Find a *drawing of* $G + F$ minimizing the edge crossings *ins*$(\boldsymbol{G}, \boldsymbol{E})$ such that the subdrawing of G is plane.

Remark. The difficulty comes from possible inequivalent embeddings of G.

3 Planar Insertion Problems

Keeping "most of" G planar...

Definition. Given a planar graph G and a set F of additional edges (vert.). Find a *drawing of* $G + F$ minimizing the edge crossings *ins*$(\boldsymbol{G}, \boldsymbol{E})$ such that the subdrawing of G is plane.

Remark. The difficulty comes from possible inequivalent embeddings of G.

Particular variants

- *Single edge insertion*: solvable in linear time using SPQR trees (easily implementable!); [**Gutwenger, Mutzel, and Weiskircher**, 2005]

3 Planar Insertion Problems

Keeping "most of" G planar...

Definition. Given a planar graph G and a set F of additional edges (vert.). Find a *drawing of* $G + F$ minimizing the edge crossings *ins*$(\boldsymbol{G}, \boldsymbol{E})$ such that the subdrawing of G is plane.

Remark. The difficulty comes from possible inequivalent embeddings of G.

Particular variants

- *Single edge insertion*: solvable in linear time using SPQR trees (easily implementable!); [**Gutwenger, Mutzel, and Weiskircher**, 2005]
- *Single vertex insertion*: solvable in polynomial time; [**Chimani, Gutwenger, Mutzel, and Wolf**, 2009]

3 Planar Insertion Problems

Keeping "most of" G planar...

Definition. Given a planar graph G and a set F of additional edges (vert.). Find a *drawing of* $G + F$ minimizing the edge crossings *ins*$(\boldsymbol{G}, \boldsymbol{E})$ such that the subdrawing of G is plane.

Remark. The difficulty comes from possible inequivalent embeddings of G.

Particular variants

- *Single edge insertion*: solvable in linear time using SPQR trees (easily implementable!); [**Gutwenger, Mutzel, and Weiskircher**, 2005]
- *Single vertex insertion*: solvable in polynomial time; [**Chimani, Gutwenger, Mutzel, and Wolf**, 2009]
- *Multiple edge insertion (MEI)*: NP-complete for general edge set F; [**Ziegler**, 2001]

3 Planar Insertion Problems

Keeping "most of" G planar...

Definition. Given a planar graph G and a set F of additional edges (vert.). Find a *drawing of* $G+F$ minimizing the edge crossings *ins*$(\boldsymbol{G},\boldsymbol{E})$ such that the subdrawing of G is plane.

Remark. The difficulty comes from possible inequivalent embeddings of G.

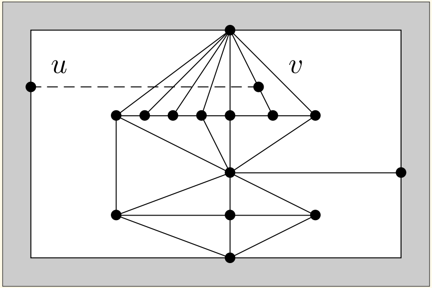
Particular variants

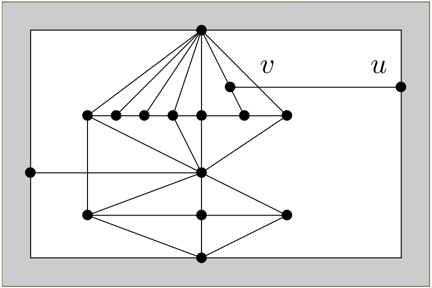
- *Single edge insertion*: solvable in linear time using SPQR trees (easily implementable!); [**Gutwenger, Mutzel, and Weiskircher**, 2005]
- *Single vertex insertion*: solvable in polynomial time; [**Chimani, Gutwenger, Mutzel, and Wolf**, 2009]
- *Multiple edge insertion (MEI)*: NP-complete for general edge set F; [**Ziegler**, 2001]

  but we may hope for a special small F... (and there are other ways)

Insertion can be very far from Crossing number

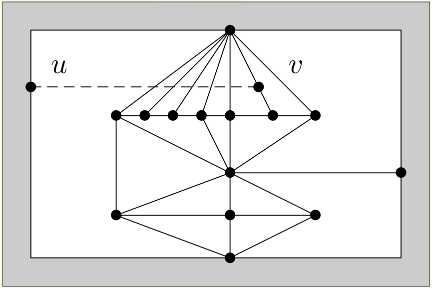
See, e.g., the following example:

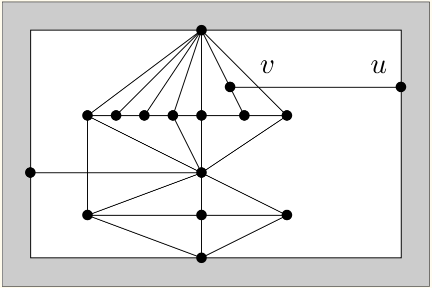




Insertion can be very far from Crossing number

See, e.g., the following example:





Remark. In cubic planar graphs, edge insertion is optimal for crossing number. [**Riskin**, 1996]

Insertion and Crossing Number

- Single edge insertion $\leftrightarrow$ *planar graph plus an edge* $G + e$

Insertion and Crossing Number

- Single edge insertion $\leftrightarrow$ *planar graph plus an edge* $G+e$
  - *cr*$(G+e)$ approximated by *ins*(G,e) up to factor $\Delta(G)$; [**PH and Salazar**, 2006]
  - factor $\lfloor \Delta(G)/2 \rfloor$, tight; [**Cabello and Mohar**, 2008]

Insertion and Crossing Number

- Single edge insertion $\leftrightarrow$ *planar graph plus an edge* $G + e$
  - *cr*$(G + e)$ approximated by *ins*(G, e) up to factor $\Delta(G)$; [**PH and Salazar**, 2006]
  - factor $\lfloor \Delta(G)/2 \rfloor$, tight; [**Cabello and Mohar**, 2008]

- Single vertex insertion $\leftrightarrow$ *apex graph* $G + x$ (specif. neighbourhood)

Insertion and Crossing Number

- Single edge insertion ↔ *planar graph plus an edge* $G + e$
  - *cr*$(G + e)$ approximated by *ins*(G, e) up to factor $\Delta(G)$; [**PH and Salazar**, 2006]
  - factor $\lfloor \Delta(G)/2 \rfloor$, tight; [**Cabello and Mohar**, 2008]
- Single vertex insertion ↔ *apex graph* $G + x$ (specif. neighbourhood)
  - *cr*$(G + x)$ approximated by *ins*(G, x) up to factor $d(x) \cdot \lfloor \Delta(G)/2 \rfloor$; [**Chimani, PH, and Mutzel**, 2008]
  - a tight factor – half of that?

Insertion and Crossing Number

- Single edge insertion $\leftrightarrow$ *planar graph plus an edge* $G+e$
  - *cr*$(G+e)$ approximated by *ins*(G,e) up to factor $\Delta(G)$;
    [**PH and Salazar**, 2006]
  - factor $\lfloor \Delta(G)/2 \rfloor$, tight; [**Cabello and Mohar**, 2008]

- Single vertex insertion $\leftrightarrow$ *apex graph* $G+x$ (specif. neighbourhood)
  - *cr*$(G+x)$ approximated by *ins*(G,x) up to factor $d(x) \cdot \lfloor \Delta(G)/2 \rfloor$;
    [**Chimani, PH, and Mutzel**, 2008]
  - a tight factor – half of that?

- Multiple edge insertion *MEI* $\leftrightarrow$ graph $G+F$ (the general case)

Insertion and Crossing Number

- Single edge insertion $\leftrightarrow$ *planar graph plus an edge* $G + e$
  - *cr*$(G + e)$ approximated by *ins*(G, e) up to factor $\Delta(G)$; [**PH and Salazar**, 2006]
  - factor $\lfloor \Delta(G)/2 \rfloor$, tight; [**Cabello and Mohar**, 2008]

- Single vertex insertion $\leftrightarrow$ *apex graph* $G + x$ (specif. neighbourhood)
  - *cr*$(G + x)$ approximated by *ins*(G, x) up to factor $d(x) \cdot \lfloor \Delta(G)/2 \rfloor$; [**Chimani, PH, and Mutzel**, 2008]
  - a tight factor – half of that?

- Multiple edge insertion *MEI* $\leftrightarrow$ graph $G + F$ (the general case)
  - *cr*$(G + F)$ approximated by *ins*(G, F) for connected planar G; the factor being $2|F| \cdot \lfloor \Delta(G)/2 \rfloor$ plus additive $\binom{|F|}{2}$ [**Chimani, PH, and Mutzel**, 2008]

Insertion and Crossing Number

- Single edge insertion $\leftrightarrow$ *planar graph plus an edge* $G+e$
  - *cr*$(G+e)$ approximated by *ins*(G,e) up to factor $\Delta(G)$;
    [**PH and Salazar**, 2006]
  - factor $\lfloor\Delta(G)/2\rfloor$, tight; [**Cabello and Mohar**, 2008]

- Single vertex insertion $\leftrightarrow$ *apex graph* $G+x$ (specif. neighbourhood)
  - *cr*$(G+x)$ approximated by *ins*(G,x) up to factor $d(x)\cdot\lfloor\Delta(G)/2\rfloor$;
    [**Chimani, PH, and Mutzel**, 2008]
  - a tight factor – half of that?

- Multiple edge insertion *MEI* $\leftrightarrow$ graph $G+F$ (the general case)
  - *cr*$(G+F)$ approximated by *ins*(G,F) for connected planar G;
    the factor being $2|F|\cdot\lfloor\Delta(G)/2\rfloor$ plus additive $\binom{|F|}{2}$
    [**Chimani, PH, and Mutzel**, 2008]
  - however, how to compute *ins*(G,F)? – enough to approximate!

4 MEI-based Approach to Crossing Numbers

Computing $\mathit{ins}(G, F)$ for planar connected G:

- [**Chuzhoy, Makarychev, and Sidiropoulos**, 2011 SODA]

  $$\leq \mathcal{O}(\Delta(G)^3 \cdot |F| \cdot \mathit{cr}(G + F) + \Delta(G)^3 \cdot |F|^2) \text{ crossings,}$$

  a very complicated algorithm for both $\mathit{cr}(G + F)$ and $\mathit{ins}(G, F)$.

4 MEI-based Approach to Crossing Numbers

Computing $\mathit{ins}(G, F)$ for planar connected G:

- [**Chuzhoy, Makarychev, and Sidiropoulos**, 2011 SODA]

  $$\leq \mathcal{O}(\Delta(G)^3 \cdot |F| \cdot \mathit{cr}(G+F) + \Delta(G)^3 \cdot |F|^2) \text{ crossings,}$$

  a very complicated algorithm for both $\mathit{cr}(G+F)$ and $\mathit{ins}(G, F)$.

- [**Chimani and PH**, 2011 ICALP]

  $$\leq \mathit{ins}(G, F) + (\lfloor \tfrac{1}{2}\Delta(G) \rfloor + \tfrac{1}{2}) \cdot (|F|^2 - |F|) \text{ crossings,}$$

  direct focus on an approximation of $\mathit{ins}(G, F)$ up to an additive factor,

4 MEI-based Approach to Crossing Numbers

Computing *ins*(G, F) for planar connected G:

- [**Chuzhoy, Makarychev, and Sidiropoulos**, 2011 SODA]

  $$\leq \mathcal{O}(\Delta(G)^3 \cdot |F| \cdot \mathit{cr}(G+F) + \Delta(G)^3 \cdot |F|^2) \text{ crossings,}$$

  a very complicated algorithm for both *cr*$(G+F)$ and *ins*(G, F).

- [**Chimani and PH**, 2011 ICALP]

  $$\leq \mathit{ins}(G,F) + (\lfloor \tfrac{1}{2}\Delta(G) \rfloor + \tfrac{1}{2}) \cdot (|F|^2 - |F|) \text{ crossings,}$$

  direct focus on an approximation of *ins*(G, F) up to an additive factor, and an easily implementable algorithm.

4 MEI-based Approach to Crossing Numbers

Computing $\mathit{ins}(G,F)$ for planar connected G:

- [**Chuzhoy, Makarychev, and Sidiropoulos**, 2011 SODA]

  $\leq \mathcal{O}(\Delta(G)^3 \cdot |F| \cdot \mathit{cr}(G+F) + \Delta(G)^3 \cdot |F|^2)$ crossings,

  a very complicated algorithm for both $\mathit{cr}(G+F)$ and $\mathit{ins}(G,F)$.

- [**Chimani and PH**, 2011 ICALP]

  $\leq \mathit{ins}(G,F) + (\lfloor \frac{1}{2}\Delta(G) \rfloor + \frac{1}{2}) \cdot (|F|^2 - |F|)$ crossings,

  direct focus on an approximation of $\mathit{ins}(G,F)$ up to an additive factor, and an easily implementable algorithm.

- For the crossing number the latter reads

  $\leq 2|F| \cdot \lfloor \frac{1}{2}\Delta(G) \rfloor \cdot \mathit{cr}(G+F) + (\lfloor \frac{1}{2}\Delta(G) \rfloor + \frac{1}{2}) \cdot (|F|^2 - |F|)$ cr.

4 MEI-based Approach to Crossing Numbers

Computing $\mathit{ins}(G, F)$ for planar connected G:

- [**Chuzhoy, Makarychev, and Sidiropoulos**, 2011 SODA]

  $$\leq \mathcal{O}(\Delta(G)^3 \cdot |F| \cdot \mathit{cr}(G + F) + \Delta(G)^3 \cdot |F|^2) \text{ crossings,}$$

  a very complicated algorithm for both $\mathit{cr}(G + F)$ and $\mathit{ins}(G, F)$.

- [**Chimani and PH**, 2011 ICALP]

  $$\leq \mathit{ins}(G, F) + (\lfloor \tfrac{1}{2}\Delta(G) \rfloor + \tfrac{1}{2}) \cdot (|F|^2 - |F|) \text{ crossings,}$$

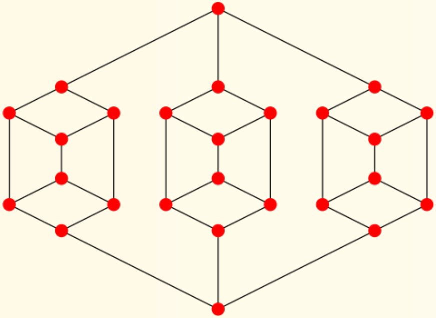
  direct focus on an approximation of $\mathit{ins}(G, F)$ up to an additive factor, and an easily implementable algorithm.

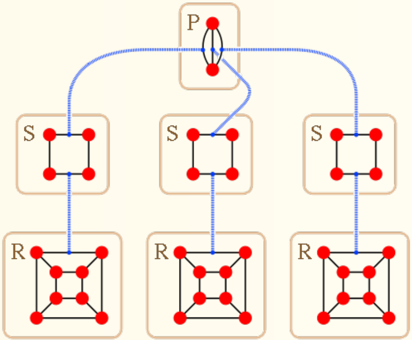
- For the crossing number the latter reads

  $$\leq 2|F| \cdot \lfloor \tfrac{1}{2}\Delta(G) \rfloor \cdot \mathit{cr}(G + F) + (\lfloor \tfrac{1}{2}\Delta(G) \rfloor + \tfrac{1}{2}) \cdot (|F|^2 - |F|) \text{ cr.}$$

So called *SPQR trees* play key role in both the approaches.

Gentle introduction to SPQR trees





- Graph broken into the *blocks* first.
- Then, for pairwise gluing on *virtual skeleton edges*, we have got
  - *S-nodes* for serial skeletons,
  - *P-nodes* for parallel skeletons,
  - *R-nodes* for 3-connected components.

5 Better Additive Approximation for MEI

Theorem. Given a conn. planar graph G and an edge set F, $F \cap E(G) = \emptyset$, the below Algorithm finds, in $\mathcal{O}(|F|^2 \cdot |V(G)|)$ time, an approximate solution to the MEI problem for G and F with

$$\leq \mathit{ins}(G,F) + 2|F| \cdot \lfloor \log_2 |F| \rfloor \cdot \lfloor \tfrac{1}{2}\Delta(G) \rfloor + \binom{|F|}{2} \text{ crossings.}$$

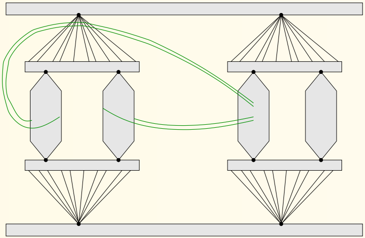
5 Better Additive Approximation for MEI

Theorem. Given a conn. planar graph G and an edge set F, $F \cap E(G) = \emptyset$, the below Algorithm finds, in $\mathcal{O}(|F|^2 \cdot |V(G)|)$ time, an approximate solution to the MEI problem for G and F with

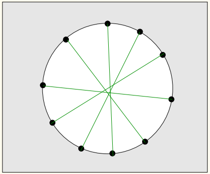
$$\leq \ ins(G,F) + 2|F| \cdot \lfloor \log_2 |F| \rfloor \cdot \lfloor \tfrac{1}{2}\Delta(G) \rfloor + \binom{|F|}{2} \quad \text{crossings.}$$

Remark. This estimate is assymptotically tight wrt. the difference between $ins(G,F)$ and the sum of individual insertions:

$|F| \cdot \log_2 |F|$



$\binom{|F|}{2}$



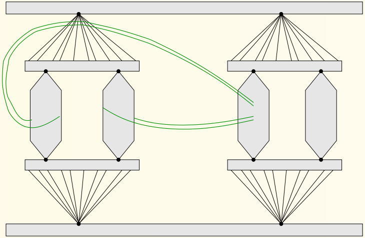
5 Better Additive Approximation for MEI

Theorem. Given a conn. planar graph G and an edge set F, $F \cap E(G) = \emptyset$, the below Algorithm finds, in $\mathcal{O}(|F|^2 \cdot |V(G)|)$ time, an approximate solution to the MEI problem for G and F with

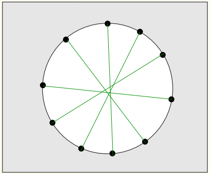
$$\leq \; \mathit{ins}(G,F) + 2|F| \cdot \lfloor \log_2 |F| \rfloor \cdot \lfloor \tfrac{1}{2}\Delta(G) \rfloor + \binom{|F|}{2} \quad \text{crossings.}$$

Remark. This estimate is assymptotically tight wrt. the difference between $\mathit{ins}(G,F)$ and the sum of individual insertions:

$|F| \cdot \log_2 |F|$



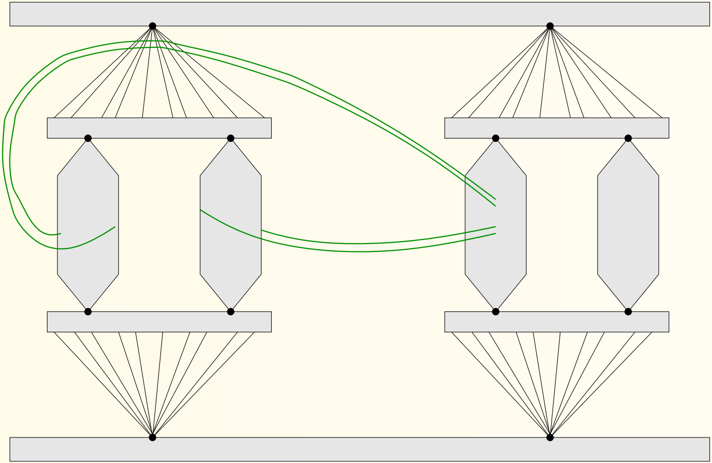
$\binom{|F|}{2}$



Corollary. The below Algorithm computes a drawing of $G+F$ with crossings

$$\leq \; 2|F| \cdot \lfloor \tfrac{1}{2}\Delta(G) \rfloor \cdot \mathit{cr}(G+F) + 2|F| \cdot \lfloor \log_2 |F| \rfloor \cdot \lfloor \tfrac{1}{2}\Delta(G) \rfloor + \binom{|F|}{2}.$$

$\Omega(|F| \cdot \log_2 |F|)$



Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,
   - more precisely, extend the insertion paths to the *block-cut tree*.

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,
   - more precisely, extend the insertion paths to the *block-cut tree*.
3. Insertion paths assign *embedding preferences* to the SPQR tree nodes;
   - combine these preferences in a "smart way" to re-embed G,

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,
   - more precisely, extend the insertion paths to the *block-cut tree*.
3. Insertion paths assign *embedding preferences* to the SPQR tree nodes;
   - combine these preferences in a "smart way" to re-embed G,
   - and insert F to G optimally (but neglecting inter-F crossings).

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,
   - more precisely, extend the insertion paths to the *block-cut tree*.
3. Insertion paths assign *embedding preferences* to the SPQR tree nodes;
   - combine these preferences in a "smart way" to re-embed G,
   - and insert F to G optimally (but neglecting inter-F crossings).
4. What happens if an embedding preference (of $\mathcal{P}_f$) is *not realized*?
   - we pay the "price" of $\lfloor \Delta(G)/2 \rfloor$ additional crossings. . .

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,
   - more precisely, extend the insertion paths to the *block-cut tree*.
3. Insertion paths assign *embedding preferences* to the SPQR tree nodes;
   - combine these preferences in a "smart way" to re-embed G,
   - and insert F to G optimally (but neglecting inter-F crossings).
4. What happens if an embedding preference (of $\mathcal{P}_f$) is *not realized*?
   - we pay the "price" of $\lfloor \Delta(G)/2 \rfloor$ additional crossings. . .
5. And how many embedding preferences are *not realized* for F?
   - any two insertion paths $\mathcal{P}_f, \mathcal{P}_g$ "divert" at $\leq$ two places, and

Very Brief Algorithm / Proof Sketch

1. Compute the *SPQR tree* of G, in linear time.
2. Optimally solve the individual edge insertions *ins*(G, f), $f \in F$;
   - each solution giving an *insertion path* $\mathcal{P}_f$ within the SPQR tree,
   - more precisely, extend the insertion paths to the *block-cut tree*.
3. Insertion paths assign *embedding preferences* to the SPQR tree nodes;
   - combine these preferences in a "smart way" to re-embed G,
   - and insert F to G optimally (but neglecting inter-F crossings).
4. What happens if an embedding preference (of $\mathcal{P}_f$) is *not realized*?
   - we pay the "price" of $\lfloor \Delta(G)/2 \rfloor$ additional crossings...
5. And how many embedding preferences are *not realized* for F?
   - any two insertion paths $\mathcal{P}_f, \mathcal{P}_g$ "divert" at $\leq$ two places, and
   - shared preferences are "the same" except at the diversions!

Embedding Preferences I

A naive approach...

The key – to define embedding preferences at SPQR-tree nodes.

Embedding Preferences I

A naive approach. . .

The key – to define embedding preferences at SPQR-tree nodes.

- The *embedding flexibility* of an SPQR-tree node:

R-node: rigid skeleton, but flip $\boxed{R}$ or $\boxed{\reflectbox{$R$}}$

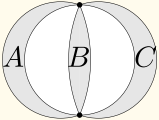
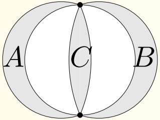
Embedding Preferences I

A naive approach. . .

The key – to define embedding preferences at SPQR-tree nodes.

- The *embedding flexibility* of an SPQR-tree node:

R-node: rigid skeleton, but flip  or 

P-node: cyclic permutation  or 

Embedding Preferences I

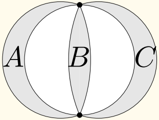
A naive approach...

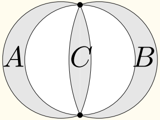
The key – to define embedding preferences at SPQR-tree nodes.

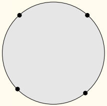
- The *embedding flexibility* of an SPQR-tree node:

R-node: rigid skeleton, but flip 

or 

P-node: cyclic permutation 

or 

S-node: just a cycle, but having two faces 

Embedding Preferences I

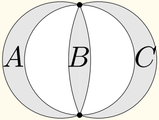
A naive approach. . .

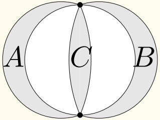
The key – to define embedding preferences at SPQR-tree nodes.

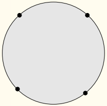
- The *embedding flexibility* of an SPQR-tree node:

R-node: rigid skeleton, but flip 

or 

P-node: cyclic permutation 

or 

S-node: just a cycle, but having two faces 

C-node: cutvertex – handled separately. . .

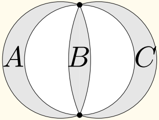
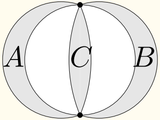
Embedding Preferences I

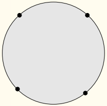
A naive approach. . .

The key – to define embedding preferences at SPQR-tree nodes.

- The *embedding flexibility* of an SPQR-tree node:

  R-node: rigid skeleton, but flip  or 

  P-node: cyclic permutation  or 

  S-node: just a cycle, but having two faces 

  C-node: cutvertex – handled separately. . .

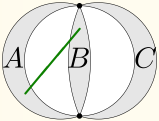
sSPQR tree – "*serialized*"; insert dummy S-nodes between all P,R nodes.

Embedding Preferences I, ctnd.

- The *embedding preference* of an SPQR-tree node (wrt. $\mathcal{P}_f$):

  P-node: "A, B together",

  where edge from A to B

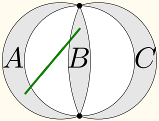


Embedding Preferences I, ctnd.

- The *embedding preference* of an SPQR-tree node (wrt. $\mathcal{P}_f$):

P-node: "A, B together",

where edge from A to B



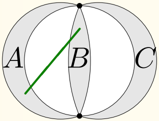
R-node: no preference (see adjacent S-nodes)

Embedding Preferences I, ctnd.

- The *embedding preference* of an SPQR-tree node (wrt. $\mathcal{P}_f$):

P-node: "A, B together",

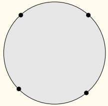
where edge from A to B



R-node: no preference (see adjacent S-nodes)

S-node: "switching" or "nonswitching",

i.e., facing different or same face of S



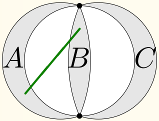
– an implicit reference to initial *default embedding* of the neighbours

Embedding Preferences I, ctnd.

- The *embedding preference* of an SPQR-tree node (wrt. $\mathcal{P}_f$):

P-node: "A, B together",

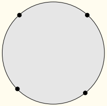
where edge from A to B



R-node: no preference (see adjacent S-nodes)

S-node: "switching" or "nonswitching",

i.e., facing different or same face of S



– an implicit reference to initial *default embedding* of the neighbours

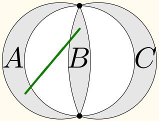
- An approximation guarantee (weak, cf. [ICALP11]):
  - at every node, the preference of some $\mathcal{P}_f$ is realized, and if another $\mathcal{P}_g$ is not satisfied there then $\mathcal{P}_f, \mathcal{P}_g$ divert there,

Embedding Preferences I, ctnd.

- The *embedding preference* of an SPQR-tree node (wrt. $\mathcal{P}_f$):

P-node: "A, B together",

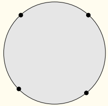
where edge from A to B



R-node: no preference (see adjacent S-nodes)

S-node: "switching" or "nonswitching",

i.e., facing different or same face of S



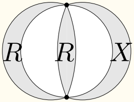
– an implicit reference to initial *default embedding* of the neighbours

- An approximation guarantee (weak, cf. [ICALP11]):
  - at every node, the preference of some $\mathcal{P}_f$ is realized, and if another $\mathcal{P}_g$ is not satisfied there then $\mathcal{P}_f, \mathcal{P}_g$ divert there,
  - consequently, at most $2\binom{|F|}{2}$ *-times paying* $\lfloor \Delta(G)/2 \rfloor$.

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;



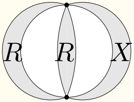
– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;

– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$



- *Default(s)* – more careful handling / specification needed;

  R-node: refer to the initial embedding (the mirror is *flipped*),

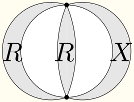
  S-node: refer to the initial "*inside*" face,

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;

– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$



- *Default(s)* – more careful handling / specification needed;

  R-node: refer to the initial embedding (the mirror is *flipped*),

  S-node: refer to the initial "*inside*" face,

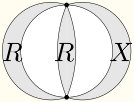
  P-node: direct the virtual (gluing) edges of the skeleton, and

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;

– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$



- *Default(s)* – more careful handling / specification needed;

  R-node: refer to the initial embedding (the mirror is *flipped*),

  S-node: refer to the initial "*inside*" face,

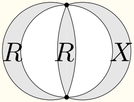
  P-node: direct the virtual (gluing) edges of the skeleton, and introduce a "*magical composition bit*" to every such virtual edge – to spec. whether the neighbour is expected to the "left/right"

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;

– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$



- *Default(s)* – more careful handling / specification needed;

  R-node: refer to the initial embedding (the mirror is *flipped*),

  S-node: refer to the initial "*inside*" face,

  P-node: direct the virtual (gluing) edges of the skeleton, and introduce a "*magical composition bit*" to every such virtual edge – to spec. whether the neighbour is expected to the "left/right"

- *Improved preferences* – the naive ones turned trully local;

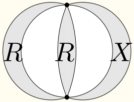
  – realization of a preference decided locally wrt. the composition bits,

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;

– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$



- *Default(s)* – more careful handling / specification needed;

  R-node: refer to the initial embedding (the mirror is *flipped*),

  S-node: refer to the initial "*inside*" face,

  P-node: direct the virtual (gluing) edges of the skeleton, and introduce a "*magical composition bit*" to every such virtual edge – to spec. whether the neighbour is expected to the "left/right"

- *Improved preferences* – the naive ones turned trully local;

  – realization of a preference decided locally wrt. the composition bits,

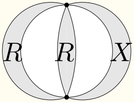
  – the composition bits then "magically disappear" ($\exists\ldots$),

Embedding Preferences II

Now making precise!

Tackle nonlocality – big hidden problem of naive preferences;

– e.g., a subpath of type $-\boldsymbol{R}-\boldsymbol{S}-\boxed{\boldsymbol{P}}-\boldsymbol{S}-\boldsymbol{R}-$



- *Default(s)* – more careful handling / specification needed;

  R-node: refer to the initial embedding (the mirror is *flipped*),

  S-node: refer to the initial "*inside*" face,

  P-node: direct the virtual (gluing) edges of the skeleton, and introduce a "*magical composition bit*" to every such virtual edge – to spec. whether the neighbour is expected to the "left/right"

- *Improved preferences* – the naive ones turned trully local;
  - realization of a preference decided locally wrt. the composition bits,
  - the composition bits then "magically disappear" ($\exists \ldots$),
  - not an easy concept, but formally very clean.

Final touch - $\log_2 |F|$

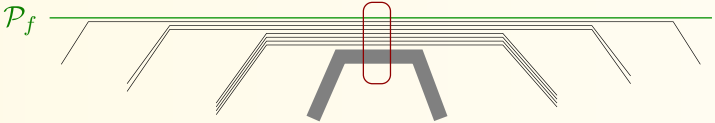
A "smart way" of combining embedding preferences in the Algorithm, plus a clever trick in the proof of the approximation guarantee...

- **Semi-majority choice of a preference** (in the Algorithm)
  - every chosen *node embedding preference* should be at least as frequent as any other one (at this node).

Final touch - $\log_2 |F|$

A “smart way” of combining embedding preferences in the Algorithm, plus a clever trick in the proof of the approximation guarantee. . .

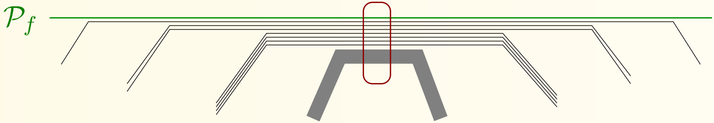
- **Semi-majority choice of a preference** (in the Algorithm)
  - every chosen *node embedding preference* should be at least as frequent as any other one (at this node).
- **“Simplicial ordering” of insertion paths** (in the Proof)
  - inductively, always take an *insertion path* $\mathcal{P}_f$ such that all other intersecting paths do so in the same node:



Final touch - $\log_2 |F|$

A "smart way" of combining embedding preferences in the Algorithm, plus a clever trick in the proof of the approximation guarantee...

- **Semi-majority choice of a preference** (in the Algorithm)
  - every chosen *node embedding preference* should be at least as frequent as any other one (at this node).
- **"Simplicial ordering" of insertion paths** (in the Proof)
  - inductively, always take an *insertion path* $\mathcal{P}_f$ such that all other intersecting paths do so in the same node:



  - then, everytime $\mathcal{P}_f$ not realized, $\geq$ half of the paths divert from $\mathcal{P}_f$.

6 Conclusions

Studying an interesting and useful MEI problem:

- A fast approximation algorithm with an additive factor.

6 Conclusions

Studying an interesting and useful MEI problem:

- A fast approximation algorithm with an additive factor.
- Nicely implementable and practically fast – see the OGDF.

6 Conclusions

Studying an interesting and useful MEI problem:

- A fast approximation algorithm with an additive factor.
- Nicely implementable and practically fast – see the OGDF.
- Hoping to get an FPT exact algorithm, wrt. $|F|$.

6 Conclusions

Studying an interesting and useful MEI problem:

- A fast approximation algorithm with an additive factor.
- Nicely implementable and practically fast – see the OGDF.
- Hoping to get an FPT exact algorithm, wrt. $|F|$.
- And, see Markus' talk...

Thank you for your attention.