Inserting Multiple Edges into a Planar Graph

Petr Hliněný

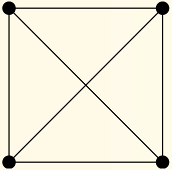
Faculty of Informatics, Masaryk University
Brno, Czech Republic

joint work with **Markus Chimani**

Osnabrück University, Germany

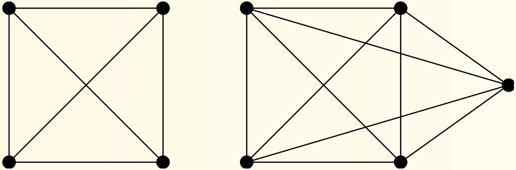
1 Drawing Graphs with Crossings

- The **crossing minimization problem**:



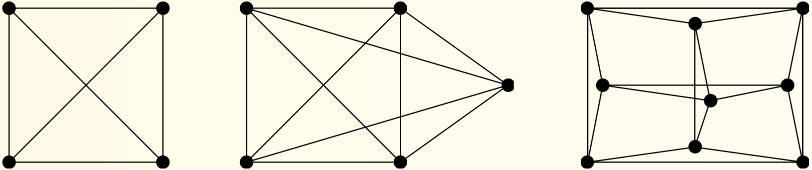
1 Drawing Graphs with Crossings

- The **crossing minimization problem**:



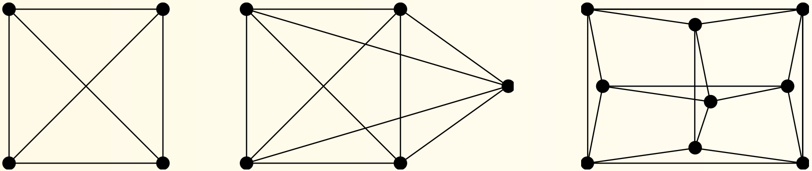
1 Drawing Graphs with Crossings

- The **crossing minimization problem**:



1 Drawing Graphs with Crossings

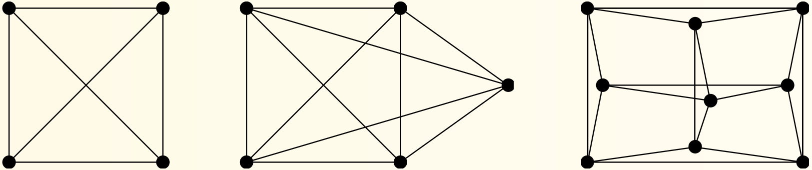
- The **crossing minimization problem**:



- *Crossing number* $cr(G) =$ the minimum number of edge crossings in G, over all possible *good drawings* of G,

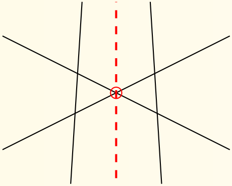
1 Drawing Graphs with Crossings

- The **crossing minimization problem**:



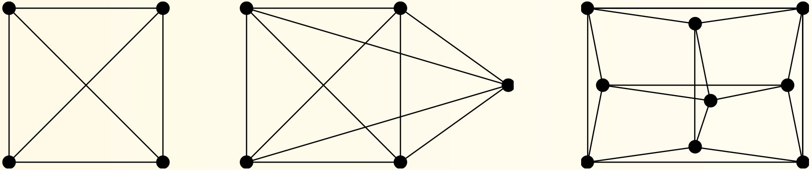
- *Crossing number* $\boldsymbol{cr(G)} =$ the minimum number of edge crossings in G, over all possible *good drawings* of G, where good means, in particular,

no

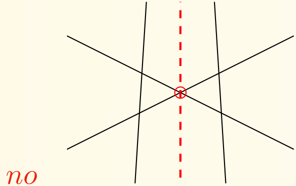


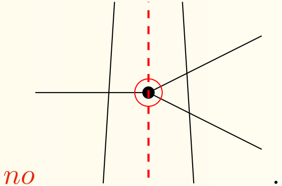
1 Drawing Graphs with Crossings

- The **crossing minimization problem**:



- *Crossing number* $\boldsymbol{cr(G)} =$ the minimum number of edge crossings in G, over all possible *good drawings* of G, where good means, in particular,



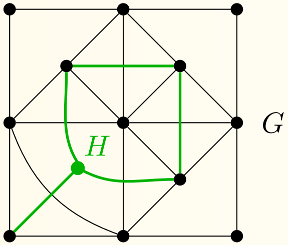


Planar Insertion Problems

Definition. Given graphs G (planar) and H, the task of *insertion of H into G* is to find a crossing-minimal drawing of $G \cup H$ such that G itself is planar in the drawing.

Planar Insertion Problems

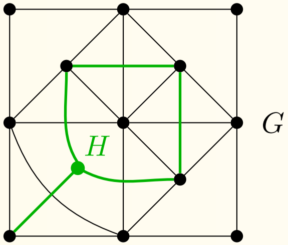
Definition. Given graphs G (planar) and H, the task of *insertion of H into G* is to find a crossing-minimal drawing of $G \cup H$ such that G itself is planar in the drawing.



– Note; $G = \emptyset \;\Rightarrow\;$ ordinary $cr(H)\ldots$

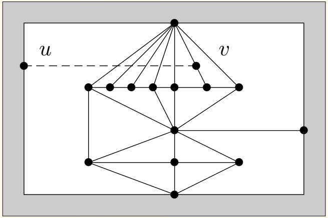
Planar Insertion Problems

Definition. Given graphs G (planar) and H, the task of *insertion of H into G* is to find a crossing-minimal drawing of $G \cup H$ such that G itself is planar in the drawing.



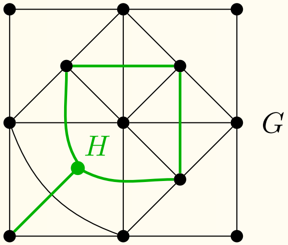
– Note; $G = \emptyset \;\Rightarrow\;$ ordinary $cr(H)\ldots$

- Optimal insertion can be very far from crossing minimization ($G + uv$):



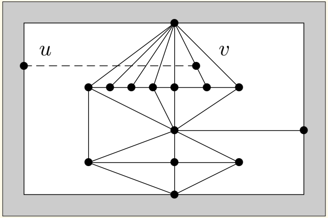
Planar Insertion Problems

Definition. Given graphs G (planar) and H, the task of *insertion of H into G* is to find a crossing-minimal drawing of $G \cup H$ such that G itself is planar in the drawing.

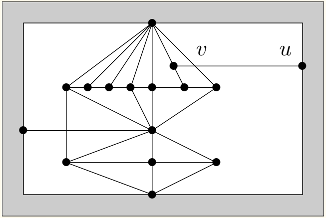


– Note; $G = \emptyset \;\Rightarrow\;$ ordinary $cr(H)\ldots$

- Optimal insertion can be very far from crossing minimization ($G + uv$):



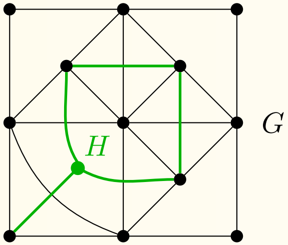
$vs.$



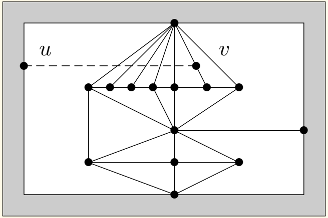
Planar Insertion Problems

Definition. Given graphs G (planar) and H, the task of *insertion of H into G* is to find a crossing-minimal drawing of $G \cup H$ such that G itself is planar in the drawing.

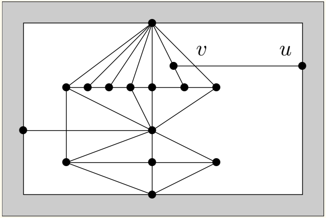
– Note; $G = \emptyset \;\Rightarrow\;$ ordinary $cr(H)\ldots$



- Optimal insertion can be very far from crossing minimization ($G + uv$):



$vs.$



- Though, sometimes useful as an approximation of the crossing number.

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

  Actually, solving insertion subproblems is the base of established crossing-number heuristics.

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

  Actually, solving insertion subproblems is the base of established crossing-number heuristics.

- There are well-studied special cases of insertion:

  – *single-edge insertion* $(H = e)$ [Gutwenger et al, 2005],

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

  Actually, solving insertion subproblems is the base of established crossing-number heuristics.

- There are well-studied special cases of insertion:
  - *single-edge insertion* $(H = e)$ [Gutwenger et al, 2005],
  - *single-vertex insertion* $(H = \text{star})$ [Chimani et al, 2009].

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

  Actually, solving insertion subproblems is the base of established crossing-number heuristics.

- There are well-studied special cases of insertion:
  - *single-edge insertion* $(H = e)$ [Gutwenger et al, 2005],
  - *single-vertex insertion* $(H = \text{star})$ [Chimani et al, 2009].

- Yet, the problem is NP-hard even with $V(H) \subseteq V(G)$ and rigid G.

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

  Actually, solving insertion subproblems is the base of established crossing-number heuristics.

- There are well-studied special cases of insertion:
  - *single-edge insertion* $(H = e)$ [Gutwenger et al, 2005],
  - *single-vertex insertion* $(H = \text{star})$ [Chimani et al, 2009].

- Yet, the problem is NP-hard even with $V(H) \subseteq V(G)$ and rigid G.

— — — —

- A bit restricted case – $V(H) \subseteq V(G)$, called *multiple-edge insertion* of $F = E(H)$, is thus a natural problem for further study.

Why insertion problems?

- Crossing minimization is very hard in general, and insertion seems easier.

  Actually, solving insertion subproblems is the base of established crossing-number heuristics.

- There are well-studied special cases of insertion:
  - *single-edge insertion* $(H = e)$ [Gutwenger et al, 2005],
  - *single-vertex insertion* $(H = \text{star})$ [Chimani et al, 2009].

- Yet, the problem is NP-hard even with $V(H) \subseteq V(G)$ and rigid G.

— — — —

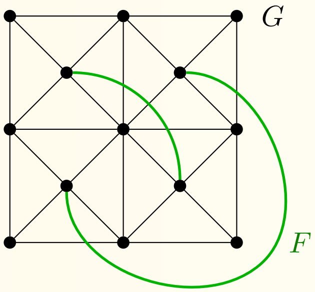
- A bit restricted case – $V(H) \subseteq V(G)$, called *multiple-edge insertion* of $F = E(H)$, is thus a natural problem for further study.

- This problem has a (practically usable!) polynomial time approximation algorithm, with only an additive error depending on $|F|$ and $\Delta(G)$.

  [Chimani and Hliněný, 2011]

2 New Contribution: Exact FPT Algorithm

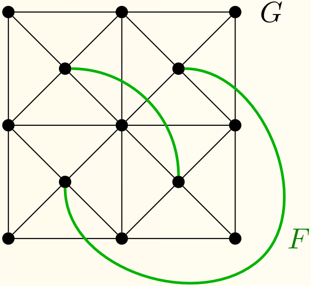
- Recalling the problem…



MEI(G, F): to find a crossing-minimal drawing of $G + F$ such that G is drawn plane.

Input: G and F
Parameter: $k = |F|$

2 New Contribution: Exact FPT Algorithm



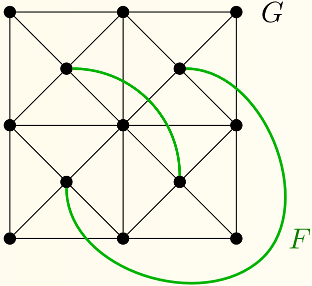
- Recalling the problem. . .

$\mathbf{MEI}(\boldsymbol{G}, \boldsymbol{F})$: to find a crossing-minimal drawing of $G + F$ such that G is drawn plane.

Input: G and F
Parameter: $k = |F|$

Theorem. Let G be a 2-connected planar graph and F a set of new edges. The $\mathrm{MEI}(G, F)$ problem is solvable to optimality in FPT time $\mathcal{O}(2^{q(k)} \cdot |V(G)|)$ where q is a polynomial.

2 New Contribution: Exact FPT Algorithm



- Recalling the problem...

MEI(G, F): to find a crossing-minimal drawing of $G + F$ such that G is drawn plane.

Input: G and F
Parameter: $k = |F|$

Theorem. Let G be a 2-connected planar graph and F a set of new edges.

The $\mathrm{MEI}(G, F)$ problem is solvable to optimality in FPT time $\mathcal{O}(2^{q(k)} \cdot |V(G)|)$ where q is a polynomial.

For connected G the same is true as long as degrees of the cutvertices of G are bounded.

Relation to previous research

- Our result directly extends the single-edge insertion algorithm [Gutwenger et al, 2005], but it is incomparable with the single-vertex insertion.

Relation to previous research

- Our result directly extends the single-edge insertion algorithm [Gutwenger et al, 2005], but it is incomparable with the single-vertex insertion.
- The crossing number problem problem $cr(G) \leq r$ is known to be in FPT with the parameter r: [Grohe, 2001] and [Kawarabayashi and Reed, 2007].

Relation to previous research

- Our result directly extends the single-edge insertion algorithm [Gutwenger et al, 2005], but it is incomparable with the single-vertex insertion.
- The crossing number problem problem $cr(G) \leq r$ is known to be in FPT with the parameter r: [Grohe, 2001] and [Kawarabayashi and Reed, 2007].

  Though, this is again incomparable to our result since

  - in one direction, even adding one edge to a planar graph may result in arbitrarily large crossing number, and
  - in the other direction, we are not able to efficiently guess which edges will be crossed ($\rightarrow F$) even if the crossing number is bounded.

Relation to previous research

- Our result directly extends the single-edge insertion algorithm [Gutwenger et al, 2005], but it is incomparable with the single-vertex insertion.
- The crossing number problem problem $cr(G) \leq r$ is known to be in FPT with the parameter r: [Grohe, 2001] and [Kawarabayashi and Reed, 2007].

  Though, this is again incomparable to our result since

  - in one direction, even adding one edge to a planar graph may result in arbitrarily large crossing number, and
  - in the other direction, we are not able to efficiently guess which edges will be crossed ($\rightarrow F$) even if the crossing number is bounded.
- Moreover, computing $cr(G + e)$ where G is planar, is NP-hard! [Cabello and Mohar, 2010]

Relation to previous research

- Our result directly extends the single-edge insertion algorithm [Gutwenger et al, 2005], but it is incomparable with the single-vertex insertion.

- The crossing number problem problem $cr(G) \leq r$ is known to be in FPT with the parameter r: [Grohe, 2001] and [Kawarabayashi and Reed, 2007].

  Though, this is again incomparable to our result since

  - in one direction, even adding one edge to a planar graph may result in arbitrarily large crossing number, and
  - in the other direction, we are not able to efficiently guess which edges will be crossed ($\rightarrow F$) even if the crossing number is bounded.

- Moreover, computing $cr(G + e)$ where G is planar, is NP-hard! [Cabello and Mohar, 2010]

- Also not comparable to prev. approximation [Chimani and Hliněný, 2011]: the approximation was polynomial-time also in $|F|$...

3 Breakdown of the Problem

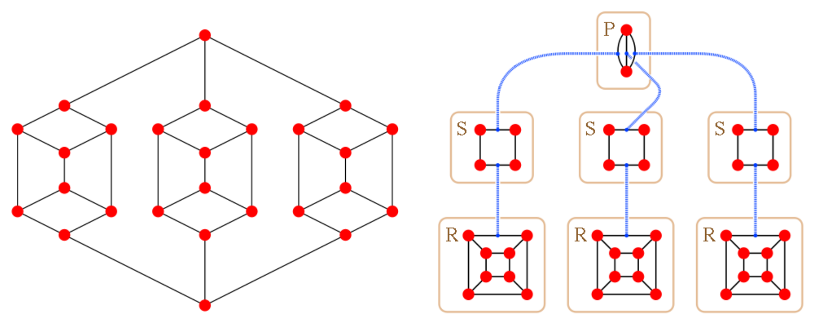
(a) G may not have a unique embedding

- Note that we cannot process all non-equivalent embeddings in FPT time.

3 Breakdown of the Problem

(a) G may not have a unique embedding

- Note that we cannot process all non-equivalent embeddings in FPT time.
- Using an established tool – so called SPQR trees:

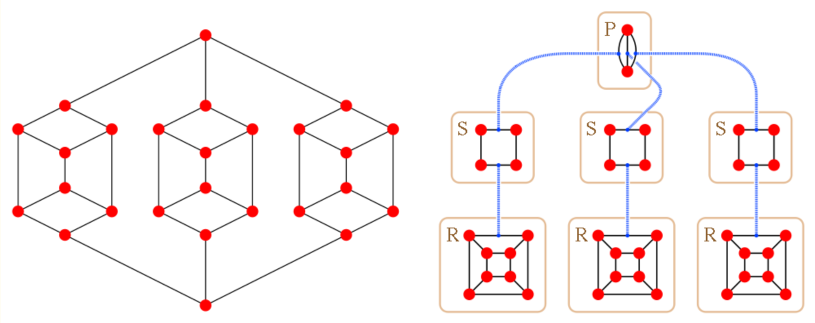


– G broken into *series, parallel, and rigid* (3-conn.) components.

3 Breakdown of the Problem

(a) G may not have a unique embedding

- Note that we cannot process all non-equivalent embeddings in FPT time.
- Using an established tool – so called SPQR trees:

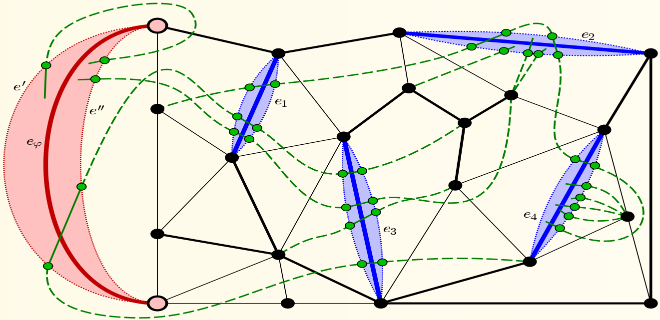


  - G broken into *series, parallel, and rigid* $(3$-conn.$)$ components.
  - Then, G is glued back together along *virtual edges*.

Dynamic programming over an SP(Q)R tree

Consider processing one SP(Q)R tree node:

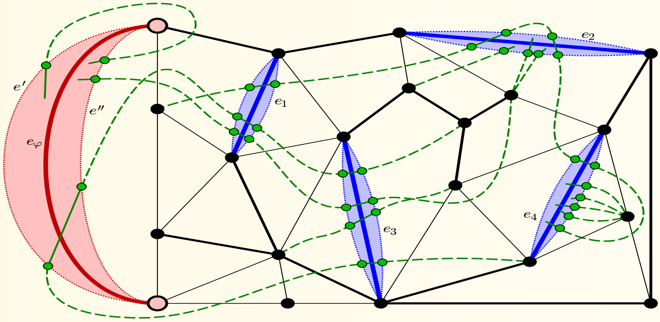
- Embedding flexibility coming from flipping components at 2-cuts.



Dynamic programming over an SP(Q)R tree

Consider processing one SP(Q)R tree node:

- Embedding flexibility coming from flipping components at 2-cuts.

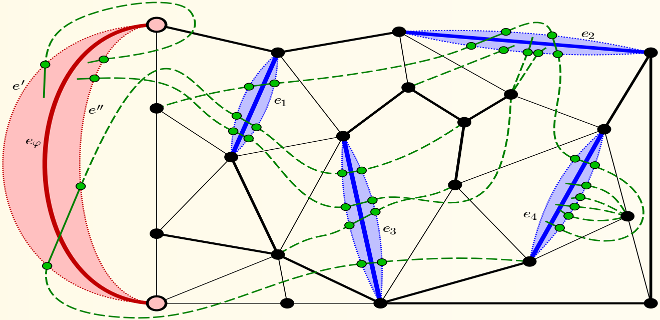


- Flipping comps. incident with edge(s) of F are *dirty* – at most $2k$ such.
  $\rightarrow$ bound the number of essential embeddings (at this node only!) in k.

Dynamic programming over an SP(Q)R tree

Consider processing one SP(Q)R tree node:

- Embedding flexibility coming from flipping components at 2-cuts.

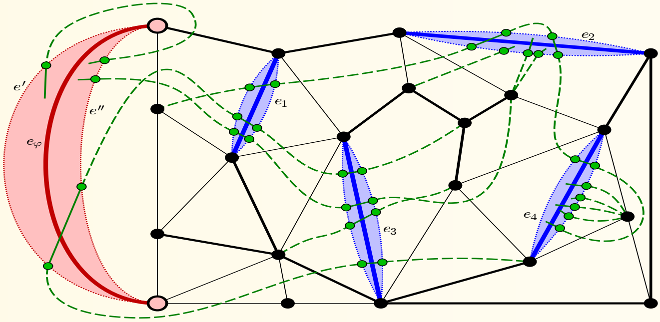


- Flipping comps. incident with edge(s) of F are *dirty* – at most $2k$ such.
  $\rightarrow$ bound the number of essential embeddings (at this node only!) in k.
- Bound the number of crossings of one flip. component as well.

Dynamic programming over an SP(Q)R tree

Consider processing one SP(Q)R tree node:

- Embedding flexibility coming from flipping components at 2-cuts.



- Flipping comps. incident with edge(s) of F are *dirty* – at most $2k$ such.
  $\rightarrow$ bound the number of essential embeddings (at this node only!) in k.
- Bound the number of crossings of one flip. component as well.
- $\Rightarrow$ At most $f(k)$ rigid cases to consider here, for some (exp.) f.

(b) G is uniquely embedded – rigid

- Generalized to cover both the primary case of 3-connected G and the rigid subcases at SPQR…

$\mathbf{r\text{-}MEI}(\boldsymbol{G_0}, \boldsymbol{F})$: given embedding G_0 stays fixed!

(b) G is uniquely embedded – rigid

• Generalized to cover both the primary case of 3-connected G and the rigid subcases at SPQR...

$\mathbf{r\text{-}MEI}(\boldsymbol{G_0, F})$: given embedding G_0 stays fixed!

– plus integer-weighted edges of G (but not F).

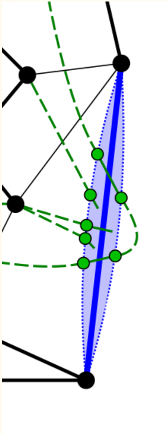
(b) G is uniquely embedded – rigid

- Generalized to cover both the primary case of 3-connected G and the rigid subcases at SPQR...

r-MEI(G_0, F): given embedding G_0 stays fixed!

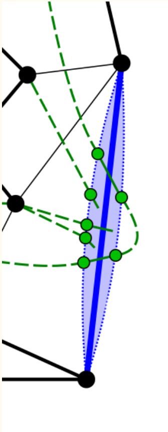
– plus integer-weighted edges of G (but not F).

- Modeling the virtual edges (flipping comps.):
  - – non-dirty $\rightarrow$ pertinent weights ($=$ edge cut),
  - – dirty ones $\rightarrow$ ∞-weight plus *connectors*.



(b) G is uniquely embedded – rigid

- Generalized to cover both the primary case of 3-connected G and the rigid subcases at SPQR…



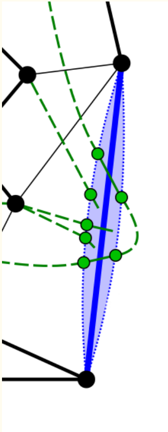
r-MEI(G_0, F): given embedding G_0 stays fixed!

– plus integer-weighted edges of G (but not F).

- Modeling the virtual edges (flipping comps.):
  - non-dirty $\rightarrow$ pertinent weights ($=$ edge cut),
  - dirty ones $\rightarrow$ ∞-weight plus *connectors*.

- Altogether, a rigid model instance with $\mathcal{O}(|V(G)|) + poly(k)$ vertices:
  - $\leq k$ F-edges, and $\leq 2k$ dirty virtual edges at this SPQR node,
  - each virtual edge crossed by an F-edge $\leq \binom{k}{2}$ times.

(b) G is uniquely embedded – rigid



- Generalized to cover both the primary case of 3-connected G and the rigid subcases at SPQR...

r-MEI(G_0, F): given embedding G_0 stays fixed!

– plus integer-weighted edges of G (but not F).

- Modeling the virtual edges (flipping comps.):
  – non-dirty $\rightarrow$ pertinent weights ($=$ edge cut),
  – dirty ones $\rightarrow$ ∞-weight plus *connectors*.

- Altogether, a rigid model instance with $\mathcal{O}(|V(G)|) + poly(k)$ vertices:
  – $\leq k$ F-edges, and $\leq 2k$ dirty virtual edges at this SPQR node,
  – each virtual edge crossed by an F-edge $\leq \binom{k}{2}$ times.

- Have to find *routes* (dual walks) for the missing segments of F-edges.

4 Solving Rigid MEI

(a) Route homotopy

(w.r.t. the ends and connectors of F-edges)

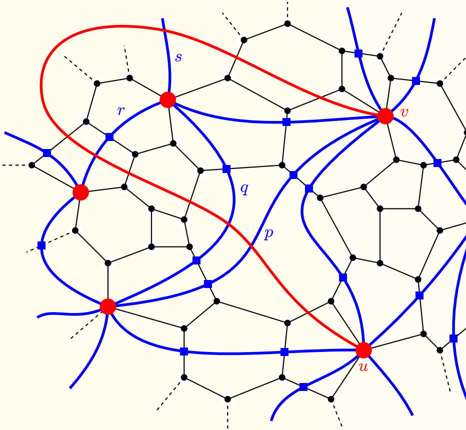
- Classical ap. – need to "triangulate" G:

4 Solving Rigid MEI

(a) Route homotopy

(w.r.t. the ends and connectors of F-edges)

- Classical ap. – need to “triangulate” G:



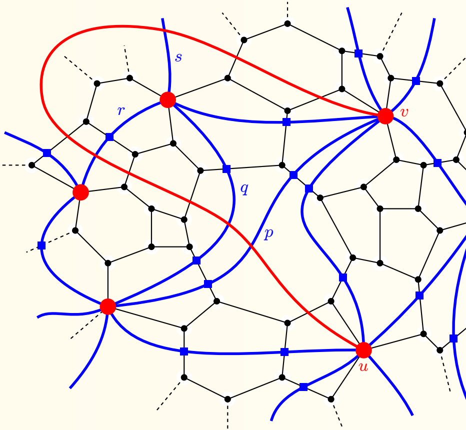
Definition. *Trinet*; $G \to (G', T)$.
Trinodes – ends (and conn.) of F-edges;
triedges – subdividing paths btw. trinodes;
altogether giving all triangular cells of T.

4 Solving Rigid MEI

(a) Route homotopy

(w.r.t. the ends and connectors of F-edges)

- Classical ap. – need to "triangulate" G:



Definition. *Trinet*; $G \to (G', T)$.
Trinodes – ends (and conn.) of F-edges;
triedges – subdividing paths btw. trinodes;
altogether giving all triangular cells of T.

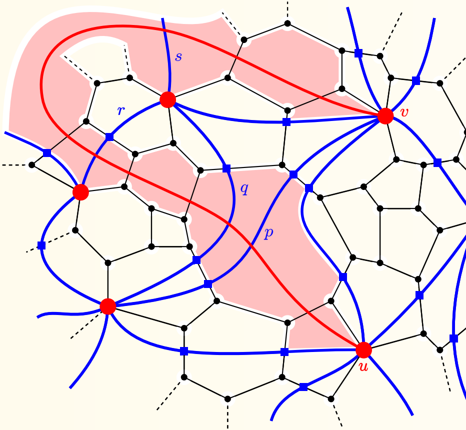
- A *shortest-spanning* trinet:
  demand the triedges to be locally
  (in part globally) shortest dual walks.

4 Solving Rigid MEI

(a) Route homotopy

(w.r.t. the ends and connectors of F-edges)

- Classical ap. – need to "triangulate" G:



Definition. *Trinet*; $G \to (G', T)$.
Trinodes – ends (and conn.) of F-edges;
triedges – subdividing paths btw. trinodes;
altogether giving all triangular cells of T.

- A *shortest-spanning* trinet:
  demand the triedges to be locally
  (in part globally) shortest dual walks.

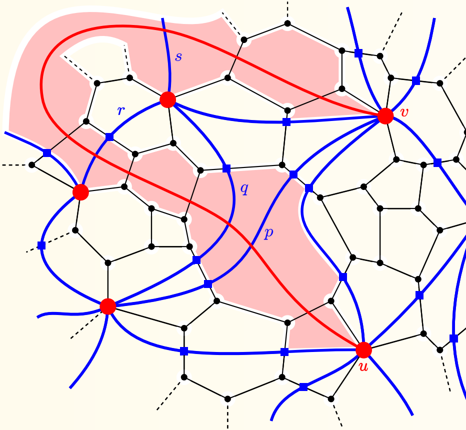
Definition. *T-sequence* over a trinet.
For $f \in F$, a sequence of intersected triedges from u to v.

4 Solving Rigid MEI

(a) Route homotopy

(w.r.t. the ends and connectors of F-edges)

- Classical ap. – need to "triangulate" G:



Definition. *Trinet*; $G \to (G', T)$.
Trinodes – ends (and conn.) of F-edges;
triedges – subdividing paths btw. trinodes;
altogether giving all triangular cells of T.

- A *shortest-spanning* trinet:
  demand the triedges to be locally
  (in part globally) shortest dual walks.

Definition. *T-sequence* over a trinet.
For $f \in F$, a sequence of intersected triedges from u to v.

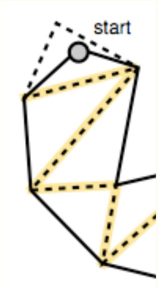
Lemma. *** In a shortest-spanning trinet, the T-sequence of an optimal $\mathrm{r\text{-}MEI}(G, F)$ solution repeats every triedge at most $8k^4$ times, where $k = |F|$.

(b) Funnel algorithm

- A straightforward adaptation to our trinets.

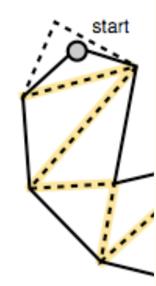
(b) Funnel algorithm

- A straightforward adaptation to our trinets.
  - only need nice “triangles” – OK,
  - and prevent switching “there and back” – loc.-shortest.
- Finding a shortest route in a sleeve simply by dual BFS.



(b) Funnel algorithm

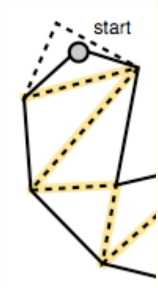
- A straightforward adaptation to our trinets.
  - only need nice "triangles" – OK,
  - and prevent switching "there and back" – loc.-shortest.
- Finding a shortest route in a sleeve simply by dual BFS.



(c) Crossing of routes

- Last to solve – when two homotopies "force" F-edges to cross each other?

(b) Funnel algorithm



- A straightforward adaptation to our trinets.
  – only need nice "triangles" – OK,
  – and prevent switching "there and back" – loc.-shortest.
- Finding a shortest route in a sleeve simply by dual BFS.

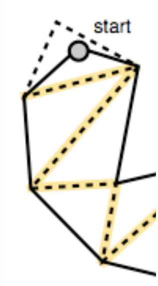
(c) Crossing of routes

- Last to solve – when two homotopies "force" F-edges to cross each other?
  $\rightarrow$ Defining a *crossing certificate* for two T-sequences.

Lemma. There exist non-crossing routes for $e, f \in F$, following T-sequences T_e, T_f, iff there is no crossing certificate for T_e, T_f.

(b) Funnel algorithm

- A straightforward adaptation to our trinets.
  - only need nice "triangles" – OK,
  - and prevent switching "there and back" – loc.-shortest.
- Finding a shortest route in a sleeve simply by dual BFS.



(c) Crossing of routes

- Last to solve – when two homotopies "force" F-edges to cross each other?

  $\rightarrow$ Defining a *crossing certificate* for two T-sequences.

Lemma. There exist non-crossing routes for $e, f \in F$, following T-sequences T_e, T_f, iff there is no crossing certificate for T_e, T_f.

- Have to similarly check also for "forcing to cross twice"...

The full "rigid" Algorithm

In: plane G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F of $w(f) = 1$.

Out: an optimal solution to (w-weighted) $\mathbf{r\text{-}MEI}(\boldsymbol{G}, \boldsymbol{F})$.

The full "rigid" Algorithm

In: plane G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F of $w(f) = 1$.

Out: an optimal solution to (w-weighted) $\mathbf{r\text{-}MEI(\boldsymbol{G}, \boldsymbol{F})}$.

1. Compute a full trinet (G', T) on the trinodes $N(T) := V(F)$, shortest-spanning;
   - globally-shortest triedges from any selected trinode to all others, and
   - then greedily add remaining triedges, each as locally-shortest.

The full "rigid" Algorithm

In: plane G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F of $w(f) = 1$.

Out: an optimal solution to (w-weighted) $\mathbf{r\text{-}MEI}(\boldsymbol{G}, \boldsymbol{F})$.

1. Compute a full trinet (G', T) on the trinodes $N(T) := V(F)$, shortest-spanning;
   - globally-shortest triedges from any selected trinode to all others, and
   - then greedily add remaining triedges, each as locally-shortest.
2. For each $f = uv \in F$; let $\mathcal{S}_f :=$ all relevant T-sequences from u to v, and
   - for $S \in \mathcal{S}_f$, compute a shortest u–v route π_S in the trinet along S.

The full “rigid” Algorithm

In: plane G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F of $w(f) = 1$.

Out: an optimal solution to (w-weighted) $\mathbf{r\text{-}MEI}(\boldsymbol{G}, \boldsymbol{F})$.

1. Compute a full trinet (G', T) on the trinodes $N(T) := V(F)$, shortest-spanning;
   - globally-shortest triedges from any selected trinode to all others, and
   - then greedily add remaining triedges, each as locally-shortest.
2. For each $f = uv \in F$; let $\mathcal{S}_f :=$ all relevant T-sequences from u to v, and
   - for $S \in \mathcal{S}_f$, compute a shortest u–v route π_S in the trinet along S.
3. For each possible system of representatives $\mathcal{P} = \{S_f\}_{f \in F}$ with $S_f \in \mathcal{S}_f$;
   - Let $X_\mathcal{P} := \big\{\{f, f'\}$: there exists a crossing certificate for $S_f, S_{f'}\big\}$
   - For $\{f, f'\} \in X_\mathcal{P}$, if two “indep.” crossing certif. of $S_f, S_{f'}$, then fail.

The full "rigid" Algorithm

In: plane G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F of $w(f) = 1$.

Out: an optimal solution to (w-weighted) $\mathbf{r\text{-}MEI}(\boldsymbol{G}, \boldsymbol{F})$.

1. Compute a full trinet (G', T) on the trinodes $N(T) := V(F)$, shortest-spanning;
   - globally-shortest triedges from any selected trinode to all others, and
   - then greedily add remaining triedges, each as locally-shortest.
2. For each $f = uv \in F$; let $\mathcal{S}_f :=$ all relevant T-sequences from u to v, and
   - for $S \in \mathcal{S}_f$, compute a shortest u–v route π_S in the trinet along S.
3. For each possible system of representatives $\mathcal{P} = \{S_f\}_{f \in F}$ with $S_f \in \mathcal{S}_f$;
   - Let $X_\mathcal{P} := \big\{\{f, f'\} : \text{there exists a crossing certificate for } S_f, S_{f'}\big\}$
   - For $\{f, f'\} \in X_\mathcal{P}$, if two "indep." crossing certif. of $S_f, S_{f'}$, then fail.
   - Otherwise, let
     $$cr_\mathcal{P} := |X_\mathcal{P}| + \sum\nolimits_{f \in F} len_w(\pi_{S_f}),$$
     where π_{S_f} is the shortest route for f and S_f, computed above.

The full “rigid” Algorithm

In: plane G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F of $w(f) = 1$.

Out: an optimal solution to (w-weighted) $\mathbf{r\text{-}MEI}(\boldsymbol{G}, \boldsymbol{F})$.

1. Compute a full trinet (G', T) on the trinodes $N(T) := V(F)$, shortest-spanning;
   - globally-shortest triedges from any selected trinode to all others, and
   - then greedily add remaining triedges, each as locally-shortest.
2. For each $f = uv \in F$; let $\mathcal{S}_f :=$ all relevant T-sequences from u to v, and
   - for $S \in \mathcal{S}_f$, compute a shortest u–v route π_S in the trinet along S.
3. For each possible system of representatives $\mathcal{P} = \{S_f\}_{f \in F}$ with $S_f \in \mathcal{S}_f$;
   - Let $X_{\mathcal{P}} := \big\{\{f, f'\}$: there exists a crossing certificate for $S_f, S_{f'}\big\}$
   - For $\{f, f'\} \in X_{\mathcal{P}}$, if two “indep.” crossing certif. of $S_f, S_{f'}$, then fail.
   - Otherwise, let
   $$\mathit{cr}_{\mathcal{P}} := |X_{\mathcal{P}}| + \sum\nolimits_{f \in F} \mathit{len}_w(\pi_{S_f}),$$
   where π_{S_f} is the shortest route for f and S_f, computed above.
4. Pick $\mathcal{P}$ with smallest $\mathit{cr}_{\mathcal{P}} < \infty$.
   Realize routing of all F-edges according to this $\mathcal{P}$, and avoid unforced crossings.

5 Final Remarks

- Handling non-2-connected G, just connected:

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

- Handling non-unit weights on the edges of F:

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

- Handling non-unit weights on the edges of F:

  We already handle edge weights on G, so why not for F?

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

- Handling non-unit weights on the edges of F:

  We already handle edge weights on G, so why not for F?
  Because the "T-sequence repetition lemma" fails with weighted F! Again subject to future investigation.

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

- Handling non-unit weights on the edges of F:

  We already handle edge weights on G, so why not for F?
  Because the "T-sequence repetition lemma" fails with weighted F! Again subject to future investigation.

- New modes of parameterization for the crossing number?

  – Known in FPT when parameterized by the solution size $cr(G)$,

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

- Handling non-unit weights on the edges of F:

  We already handle edge weights on G, so why not for F?
  Because the "T-sequence repetition lemma" fails with weighted F! Again subject to future investigation.

- New modes of parameterization for the crossing number?

  – Known in FPT when parameterized by the solution size $cr(G)$,

  – but what if we parameterize by the number of edges which "cover" all the crossings?

5 Final Remarks

- Handling non-2-connected G, just connected:

  The are problems with cutvertices of high degree – cannot enumerate possible rigid subcases in FPT, but subject to ongoing investigation.

- Handling non-unit weights on the edges of F:

  We already handle edge weights on G, so why not for F?
  Because the "T-sequence repetition lemma" fails with weighted F! Again subject to future investigation.

- New modes of parameterization for the crossing number?

  - Known in FPT when parameterized by the solution size $cr(G)$,
  - but what if we parameterize by the number of edges which "cover" all the crossings?

Thank you for your attention.