I/O-Efficient Algorithms on Near-Planar Graphs

LATIN 2006
Valdivia, Chile

Herman Haverkort
Eindhoven University, NL

Laura Toma
Bowdoin College, USA

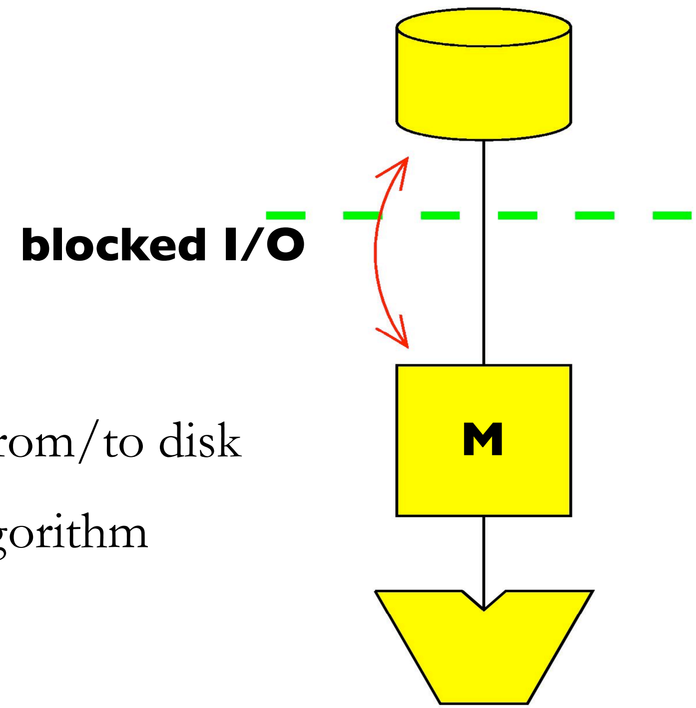
I/O-Efficient Graph Algorithms: Motivation

- Massive graphs
  - GIS (Geographic Information Systems)
    - Terabytes of data; e.g. NASA SRTM project, LIDAR data
    - Terrains modeled as triangulations, contour lines, or grids (graphs)
    - TIGER road data
  - Internet graph
  - Physics, astronomy

- On massive graphs the bottleneck is usually the I/O

I/O-Efficient Algorithms

- Graph G = (V, E) stored on disk
- I/O-model [AV'88]
  - M = main memory size
  - B = disk block size
  - I/O operation= reading/writing a block of data from/to disk
- I/O-efficiency: number of I/Os performed by the algorithm
- Basic I/O bounds
  - Scanning: $scan(E) = \Theta\left(\frac{E}{B}\right)$
  - Sorting: $sort(E) = \Theta\left(\frac{E}{B}\log_{M/B}\frac{E}{B}\right)$
- In practice M and B are big: scan(E) < sort(E) $\ll$ E I/Os



I/O-Efficient Algorithms: Related work

Lower bound: $\Omega(\min(V, sort(V)))$

- practically $\Omega(sort(V))$

General directed graphs

- BFS, SSSP, DFS: $\Theta\left((V + \frac{E}{B}) \lg V + sort(E)\right)$ [BVWB'00]
- sparse $(E = O(V)) \Longrightarrow \Omega(V)$

General undirected graphs:

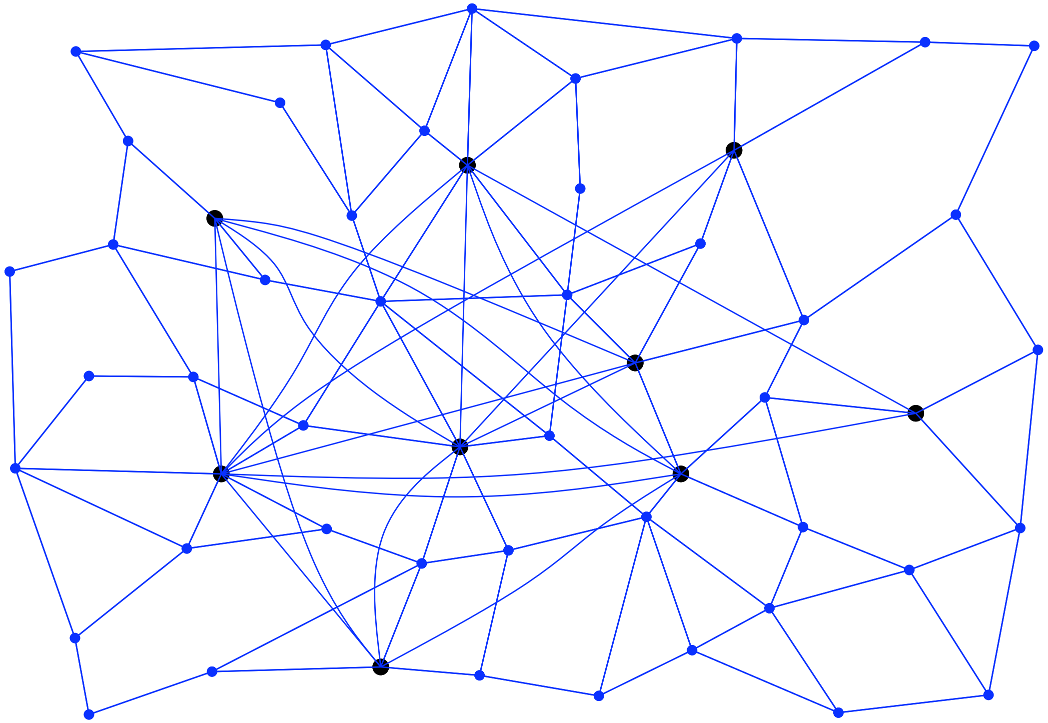
- SSSP: $\Theta\left(V + \frac{E}{B} \lg V\right)$ [KS'96]; $\Theta\left(\sqrt{\frac{VE}{B} \lg \frac{W}{w}} + sort(E)\right)$ [MZ'03]
- sparse $(E = O(V)) \Longrightarrow \Omega(V)$ $(\Omega(\frac{V}{\sqrt{B}})$ if bounded weights)

Planar directed graphs:

- SSSP: $\Theta(sort(V))$ [ATZ'03]
- DFS: $\Theta(sort(V) \lg V)$ [AZ'03]

Motivation

- Planar directed graphs
  - SSSP: $\Theta(\text{sort}(V))$
- Sparse directed graphs
  - SSSP: $\Omega(V)$
- Lower bound: practically $\Omega(\text{sort}(V))$



Our Results

Let $G = (V, E \cup E_C)$, where $\mathcal{K} = (V, E)$ is planar and $G_C = G - \mathcal{K} = (V_C, E_C)$ is the non-planar part of G, given.

- We show how to find small separators for G that gracefully depend on the non-planar part of G
- Compute SSSP in $O(E_C + sort(V + E_C))$ I/Os.
- Generalize to graphs $G = (V, E \cup E_C)$ s.th. $\mathcal{K}$ can be drawn with T crossings. SSSP in $O(E_C + sort(V + T + E_C))$ I/Os.
- Obtain similar results for BFS, DFS, topological order and conn. comp.

Near-planar graphs: If $T = O(V)$ and $E_C = O(V/B)$:

- SSSP, BFS, CC, topological order in $O(sort(V))$, DFS in $O(\frac{V}{\sqrt{B}})$ I/Os.

If a suitable drawing of $G = (V, E)$ is given, SSSP can be computed in $O(sort(E))$ on graphs with low crossing number, graphs that are k-embeddable in the plane, graphe with low skewness and graphs with low splitting number.

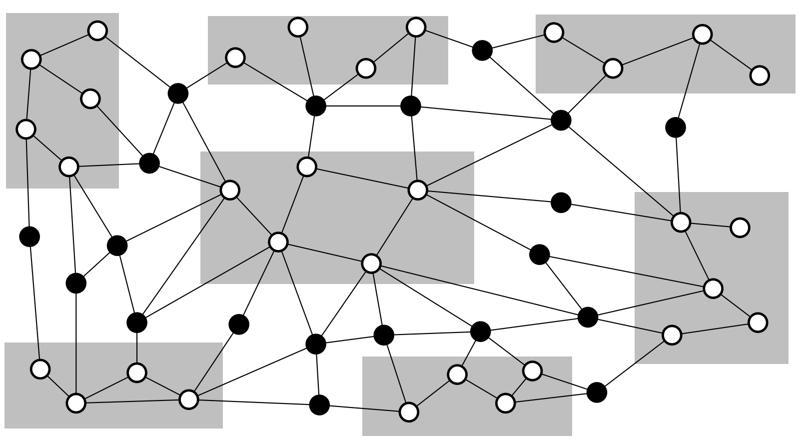
Outline

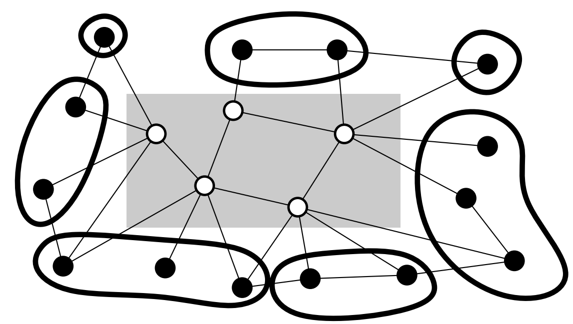
- Partitioning a planar graph
- General approach to I/O-efficient SSSP using a partition
- Partitioning a near-planar graph
- SSSP using a near-planar partition
- Planarizing graphs
- Discussion and open questions

Partitioning a Planar Graph

R-partition: For any R, a planar graph $\mathcal{K} = (V, E)$ can be partitioned using a set V_S of separator vertices into subgraphs (clusters) $\mathcal{K}_i$ such that

- Each cluster $\mathcal{K}_i$ has at most $O(R)$ vertices
- There are $O(\frac{V}{R})$ clusters in total
- The number of separator vertices is $O(\frac{V}{\sqrt{R}})$
- Each cluster $\mathcal{K}_i$ is adjacent to $O(\sqrt{R})$ separator vertices



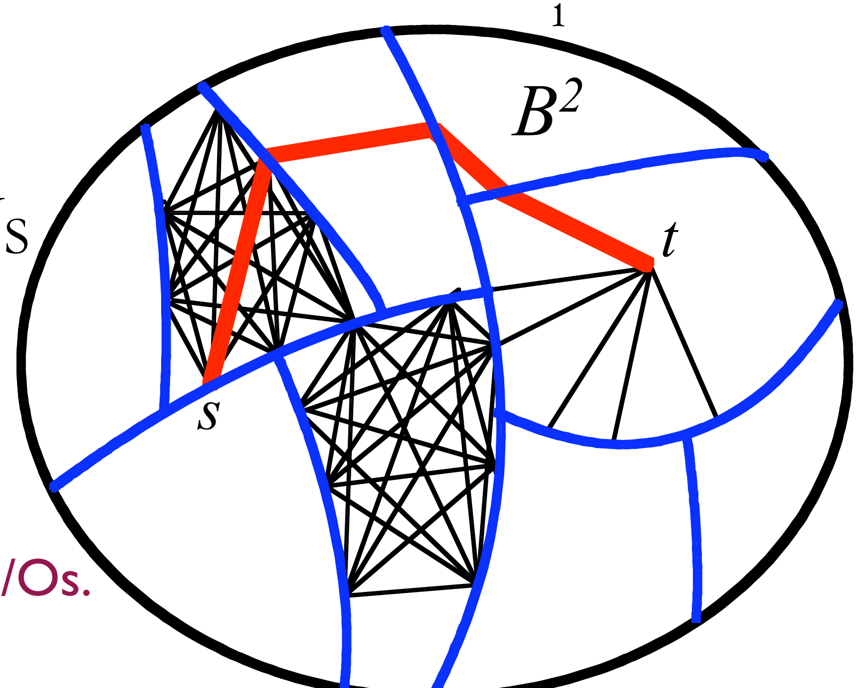


cluster $\mathcal{K}_i$ and its *boundary* $\delta\mathcal{K}_i$
(the set of separator vertices adj to $\mathcal{K}_i$)

Boundary set: a maximal set of separator vertices adjacent to same clusters.
Lemma: The number of boundary sets in an R-partition is $O(\frac{V}{R})$.

I/O-Efficient Planar SSSP

- Compute a B^2-partition of K
- Construct a substitute graph K^R on the separator vertices with the property that it preserves SP in K between any u,v in V_S
  - How? Replace each cluster with a complete graph on its boundary. For any u,v on the boundary of K_i, the weight of edge (u,v) in K^R is ∂_{K_i} (u,v)
- Solve SSSP on K^R
  - This gives SP in K to all vertices in V_S
- Find SP to vertices inside clusters



Theorem:
Planar SSSP can be solved in O(sort(V)) I/Os.

Partitioning a Near-Planar Graph

- Notation
  - $G = (V, E \cup E_C)$
  - $K = (V, E)$ **planar**

    cross-link vertices

    cross-link edges
  - $G_C = G - K = (V_C, E_C)$ **cross-link** part of G
- Compute an R-partition of K
  - V_S not a separator for G
- Add all cross-link vertices to V_S ? Planar SSSP takes $O(V + E_C)$ I/Os
- Goal: refine partition to include V_C and bound the number of cross-link vertices per cluster

Partitioning a Near-Planar Graph

Refine subgraphs that contain more than $c\sqrt{R}$ cross-link vertices

In the paper we prove the following:

Lemma: Given a subgraph $G = (V, E)$ of a planar graph with $|\delta G| = O(\sqrt{V})$, and a weight function $w : V \to \mathcal{R}$ such that $\sum_{v \in V} w(v) = W$, we can find a subset $S \subset V$ of size $O(\sqrt{VW})$ which separates $G - S$ into a set of $O(W)$ subgraphs (clusters) G' with the following properties:

- each cluster $G' = (V', E')$ has a total weight $\sum_{v \in V'} w(v)$ of at most 1.
- for each cluster $G' = (V', E')$, we have that $\partial G'$ has $O(\sqrt{V})$ vertices.

Apply lemma to every subgraph $\mathcal{K}_i$ that has more than $c\sqrt{R}$ cross link vertices

- assign $w = 1/(c\sqrt{R})$ to $v \in V_C$ and $w = 0$ otherwise

Thus every refined subraph has $O(R)$ vertices, $O(\sqrt{R})$ cross-link vertices and $O(\sqrt{R})$ on its boundary.

Partitioning a Near-Planar Graph

Overall we prove the following:

For any graph $G = (V, E \cup E_C)$ and any R we can find a subset $V_S \subset V$ whose removal separates $\mathcal{K}$ into a set of subgraphs G_i with the following properties:

- the total number of vertices in V_S is $O(V/\sqrt{R} + \sqrt{VV_C}/R^{1/4})$
- there are $O(V/R + V_C/\sqrt{R})$ subgraphs G_i in $\mathcal{K} - V_S$
- each subgraph contains $O(R)$ vertices, is adjacent to $O(\sqrt{R})$ separator vertices and contains $O(\sqrt{R})$ cross-link vertices

This refined partition can be computed with $sort(E)$ I/Os.

SSSP using a Refined R-Partition

- Compute a refined R-partition ($R = B^2$)
  - A SP can enter and exit a cluster through a separator or cross-link vertex
- Compute a substitute graph G^R on the separator and cross-link vertices with the property that it preserves SP in G between any u,v in $V_S \cup V_C$
  - G^R contains G_C and the subgraph induced by Vs
  - Replace each cluster with a complete graph on its boundary and cross-link vertices. For any u,v on ∂Gi, weight of (u,v) is ∂_{G_i} (u,v)
- Compute SSSP on G^R
  - This gives SP in G to all vertices in $V_S \cup V_C$
- Find SP to vertices inside clusters

SSSP using a Refined R-Partition

- Compute G^R
  - G^R contains G_C and the subgraph induced by V_S $\Rightarrow O(V_S + V_C)$
  - Replace each cluster with a complete graph on its boundary and cross-link vertices $\Rightarrow$ each subgraph contributes O(R) edges
  - Lemma: G^R has $O(V/\sqrt{R}+\sqrt{VV_C}/R^{1/4}+V_C)$ vertices and $O(V+V_C\sqrt{R}+E_C)$ edges and can be computed in $O(scan(E)+sort(|G^R|))$ I/Os.
- Compute SSSP on G^R
  - Use Dijkstra's algorithm and I/O-efficient priority queue
  - Keep a list L of current distances from to all vertices in G^R ; use L throughout Dijkstra to read and update the distances to neighbor vertices
  - But.. we cannot afford one I/O per edge. Store L as follows: all $v \in V_S$ grouped by boundary set followed by all $v \in V_C - V_S$ grouped by cluster
- Compute SP to vertices inside clusters:
  - Load each cluster and its boundary in memory

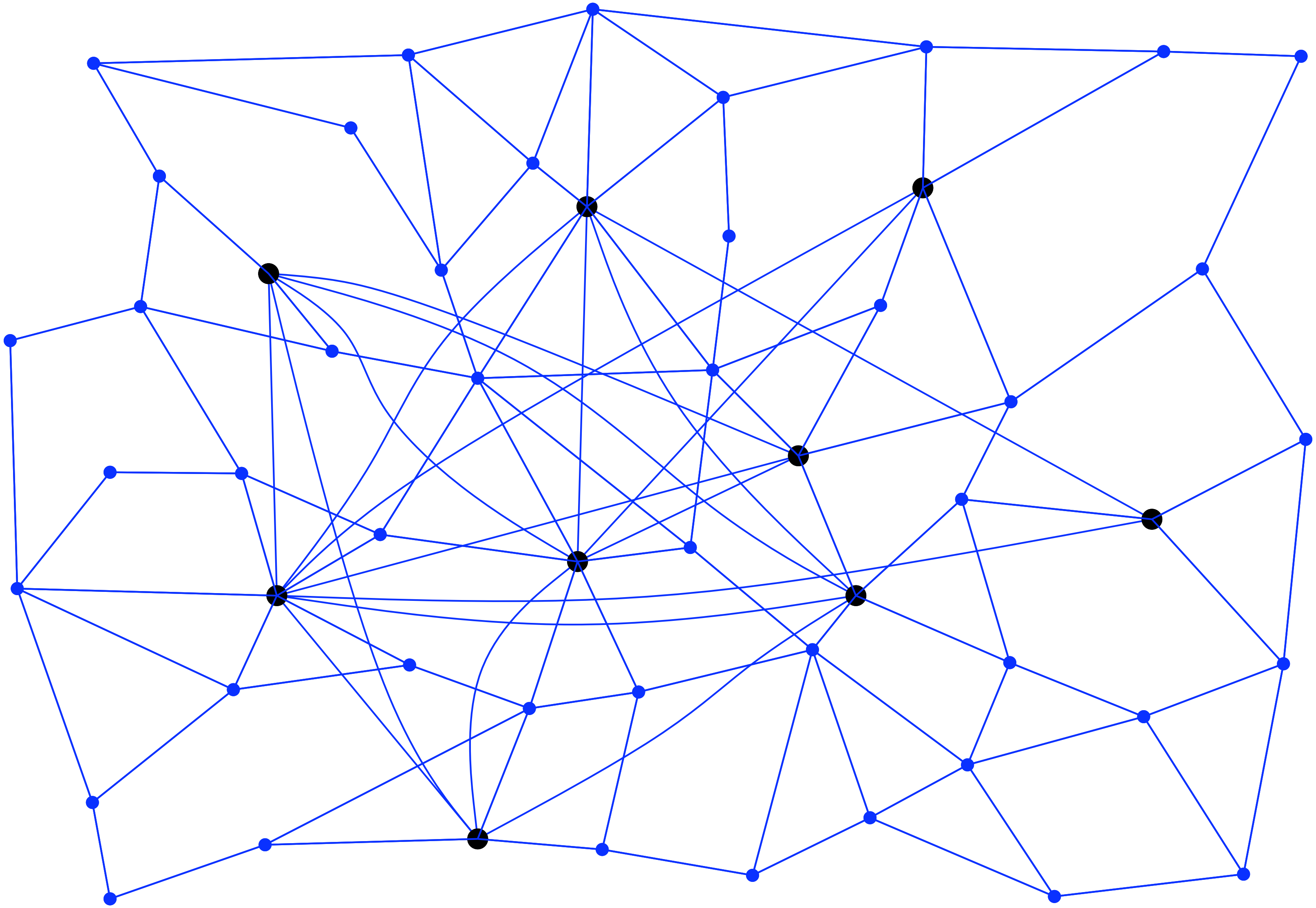
Results

SSSP on a digraph $G = \mathcal{K} \cup G_C$ uses $O(E_C + sort(V + E_C))$ I/Os.

- The ideas can be extended to
  - Connected components (assuming G is undirected)
  - Topological order (assuming G is acyclic)
  - DFS

Topological order and the connected components of G can be computed with $O(E_C + sort(V + E_C))$ I/Os.

A DFS ordering can be computed with $O(V/\sqrt{B} + E_C)$ I/Os.

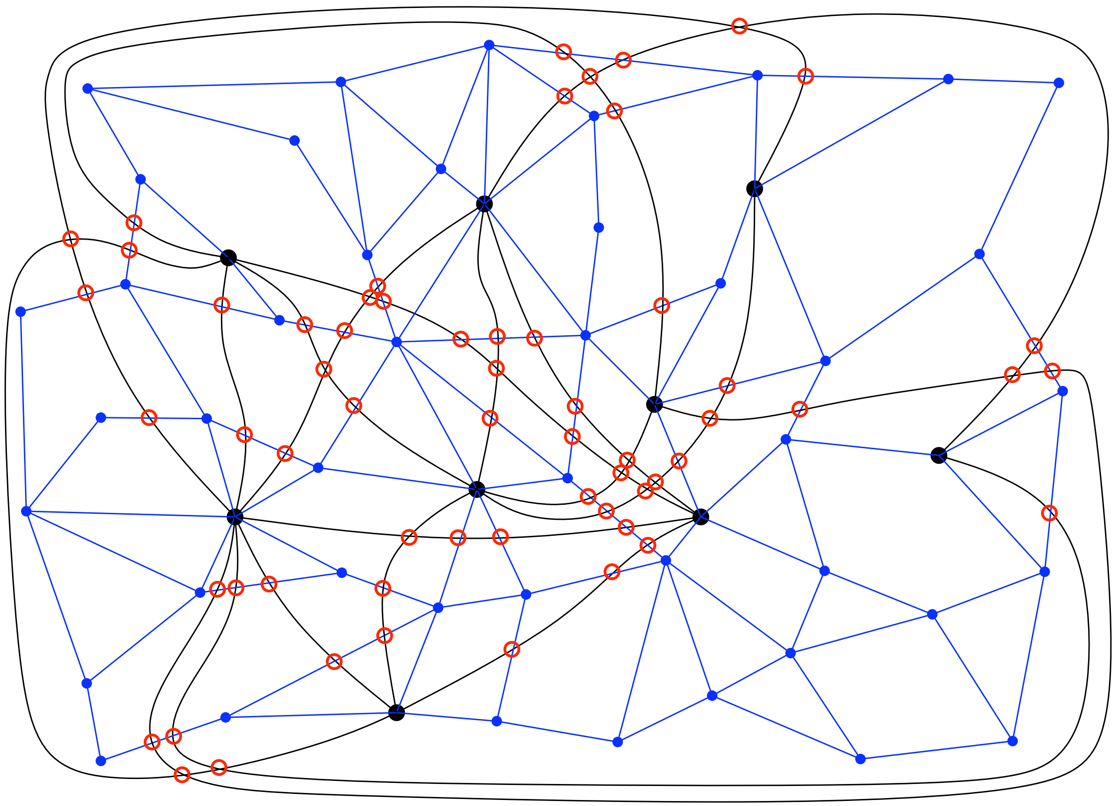


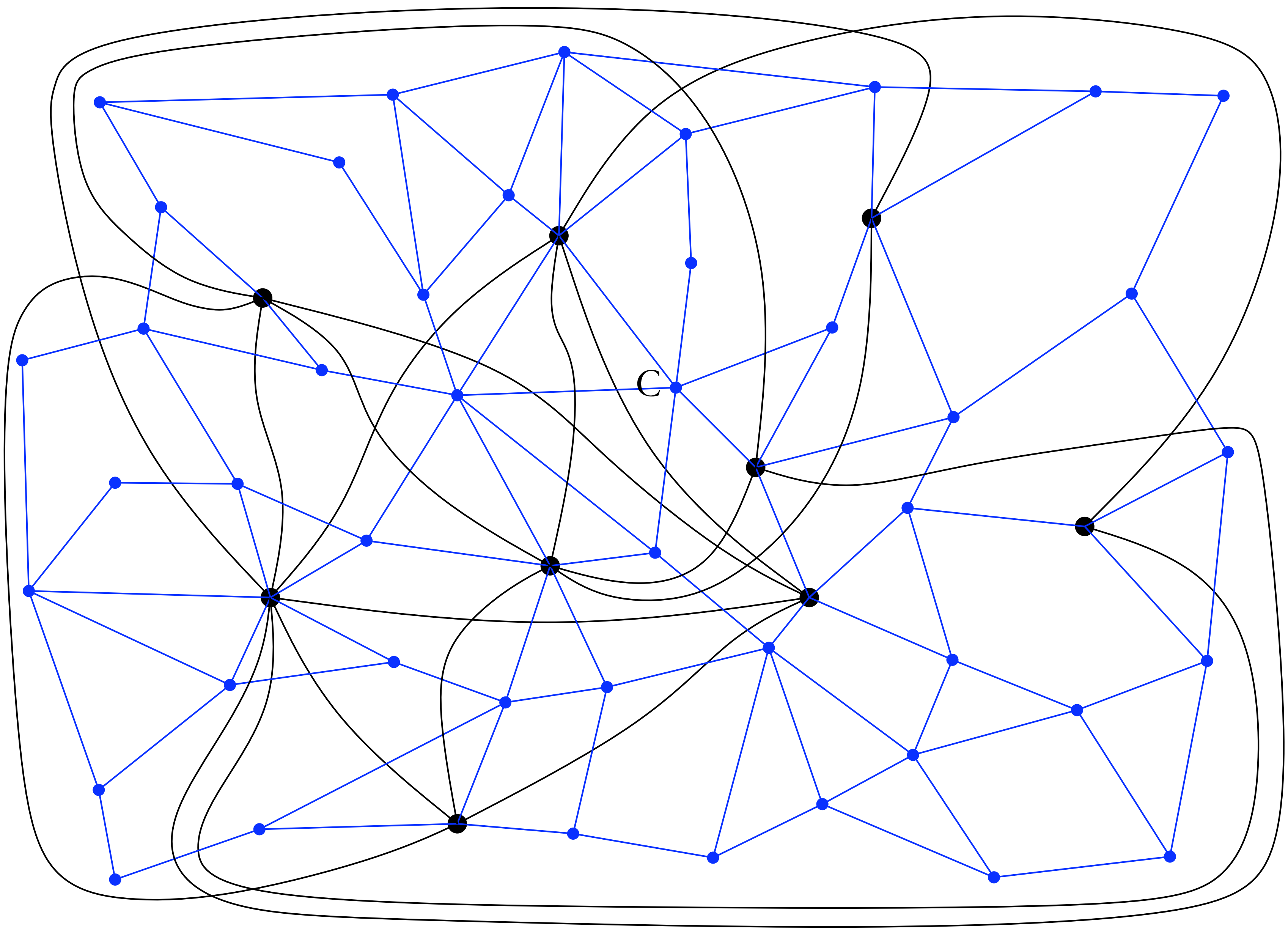
Planarizing G

- The algorithms assume G is given as $G = (V, E \cup E_C)$, $K = (V, E)$ **planar**
- How to find K?
- Measures of planarity [Liebers JACM 2001]
  - Crossing number
  - k-embeddability
  - Skewness
  - Splitting number

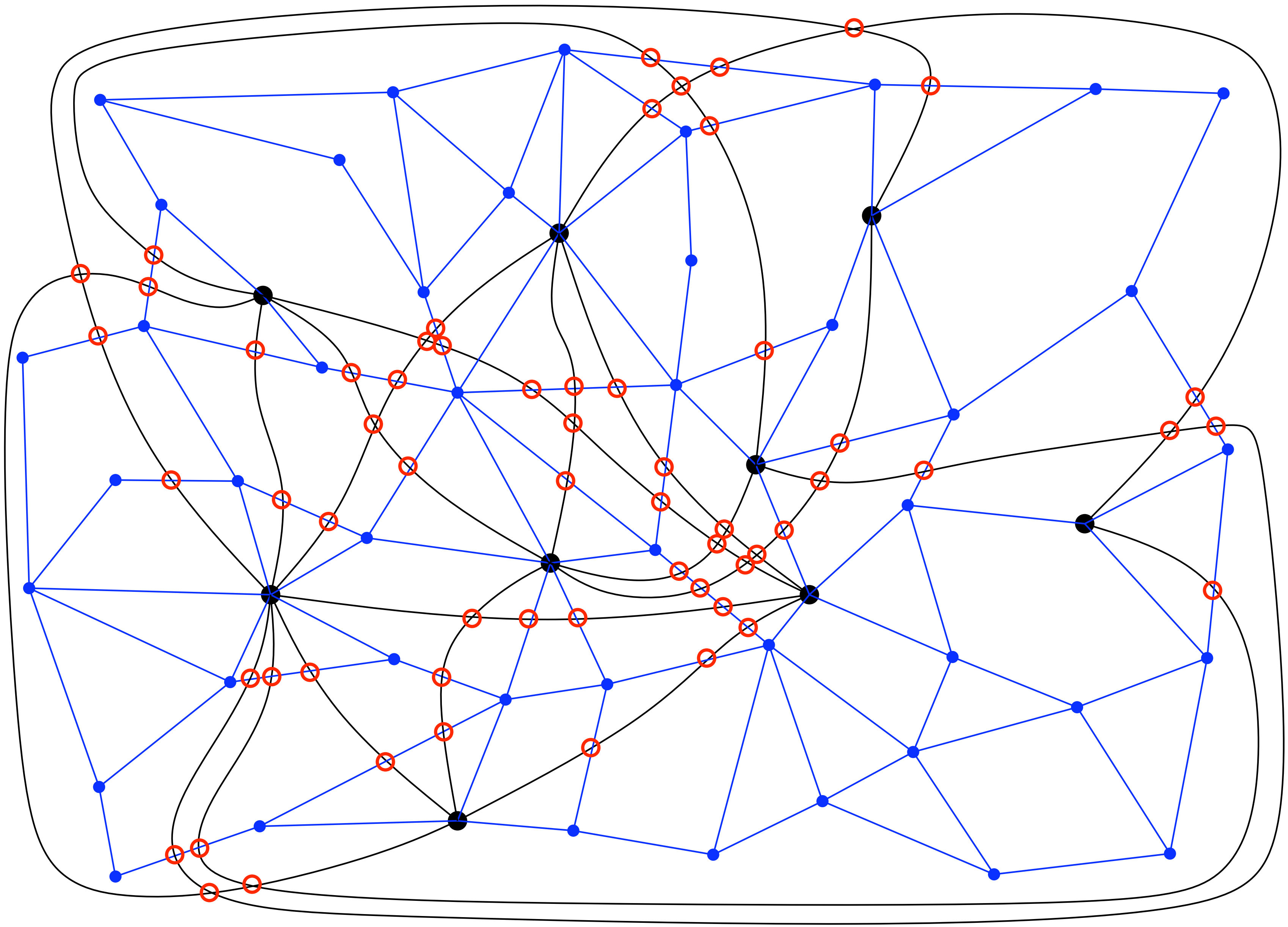
Graphs with Low Crossing Number

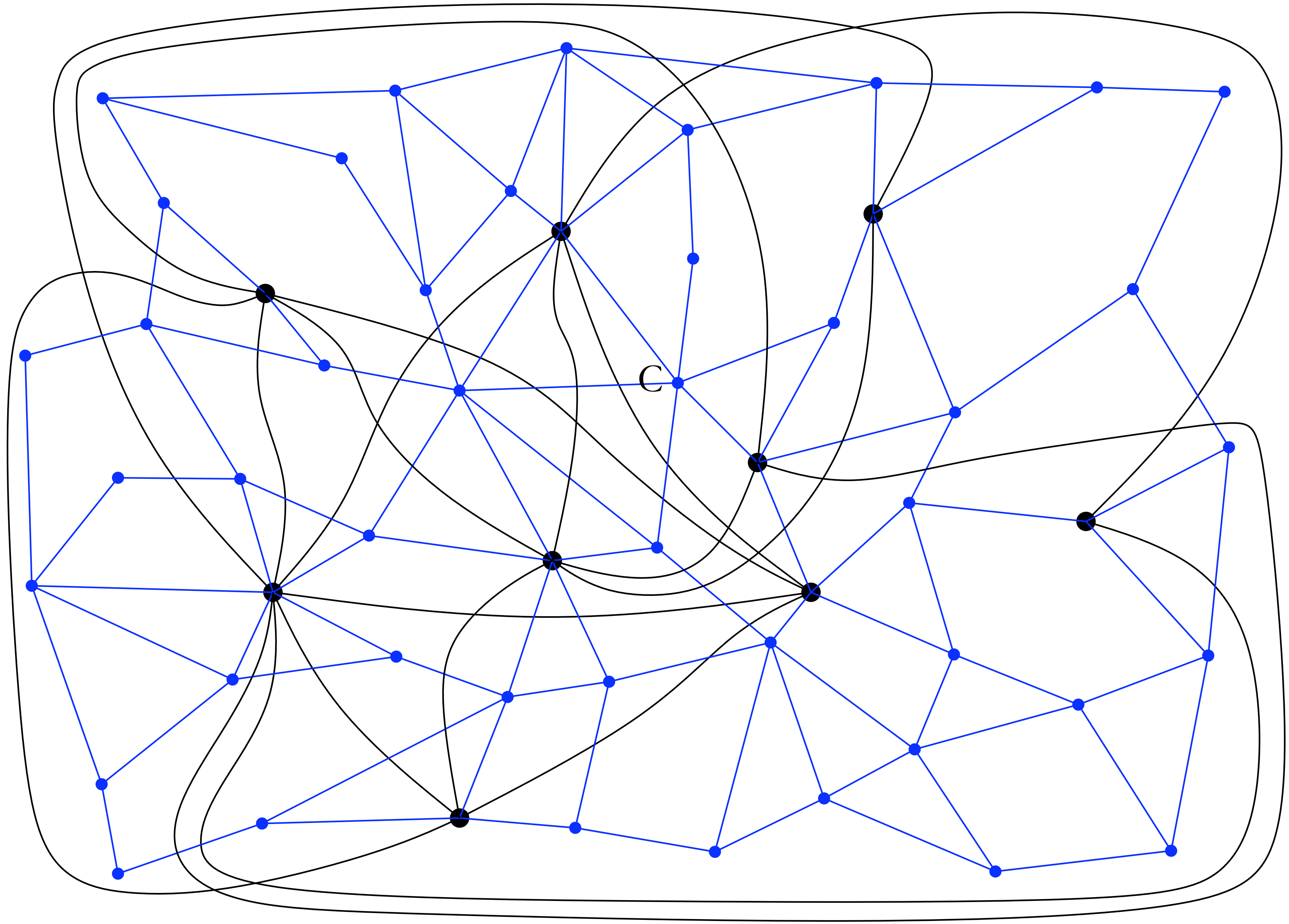
- Crossing number - minimum nb of edge crossings needed in any drawing of G in the plane
- Finding crossing nb of G is NP-complete
- When a drawing of G=(V,E) with T crossing is given ⇒ preprocess G to solve SSSP in O(sort(E+T)) I/Os
    - Represent each crossing by a vertex

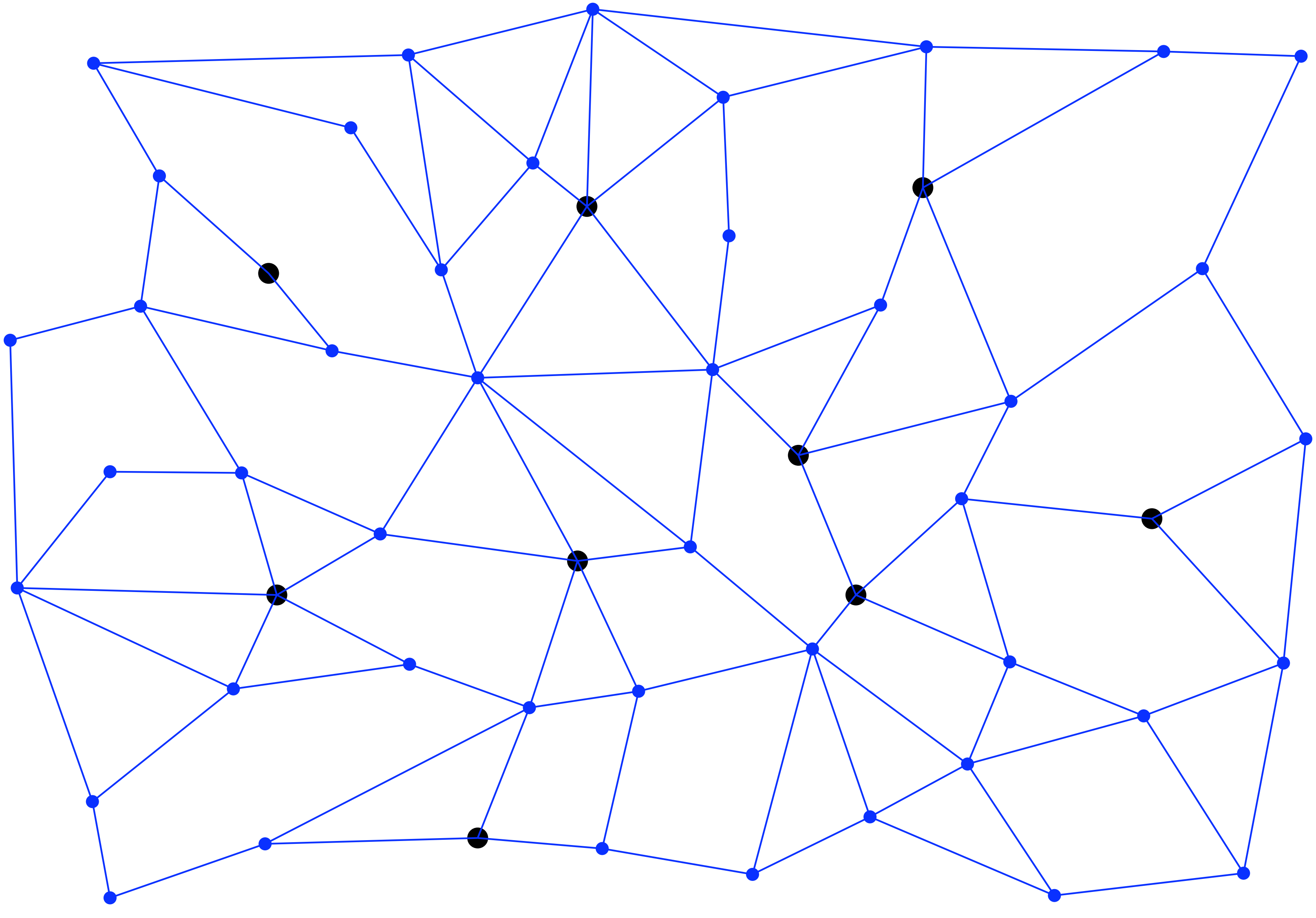



C

61 crossings 17 cross edges




C



Graphs with Low Skewness

- Skewness of $G=(V, E)$ is the min size of any set of edges E_C s.t. $G-E_C$ is planar
- Finding skewness of G is NP-complete (finding maximum planar subgraph)

- If E_C given and $E_C = O(E/B) \Rightarrow$ SSSP in $O(E_C + sort(E))$
  - The crossing number could be large
- If a drawing of G is given
  - Define a crossing graph G'=(V',E'): G' has a vertex v(e) for every edge e in G; and an edge (v(e), v(f)) for every pair of crossing edges e and f in G
  - A maximum matching in G' gives a 2-approximation of a min set E_C such that $G-E_C$ is intersection -free
  - Compute a matching of G' in O(sort(E'))= O(sort(T)), T = nb crossings in G [ABW'02]

Conclusion, Open Questions

- Extend I/O-efficient algorithms to graphs that are near-planar
  - Graphs with low crossing number, low skewness, low splitting number
- Our algorithms can handle such graphs if a suitable drawing is given
- SSSP in O(sort(E))
  - CC, topological sort, DFS
- If drawing is not given, identifying MPG is NP-complete

- Questions
  - Other measures of planarity (thickness)
  - Constant-size approx for finding cross-links with O(E) crossings