Configurations with Few Crossings in Topological Graphs

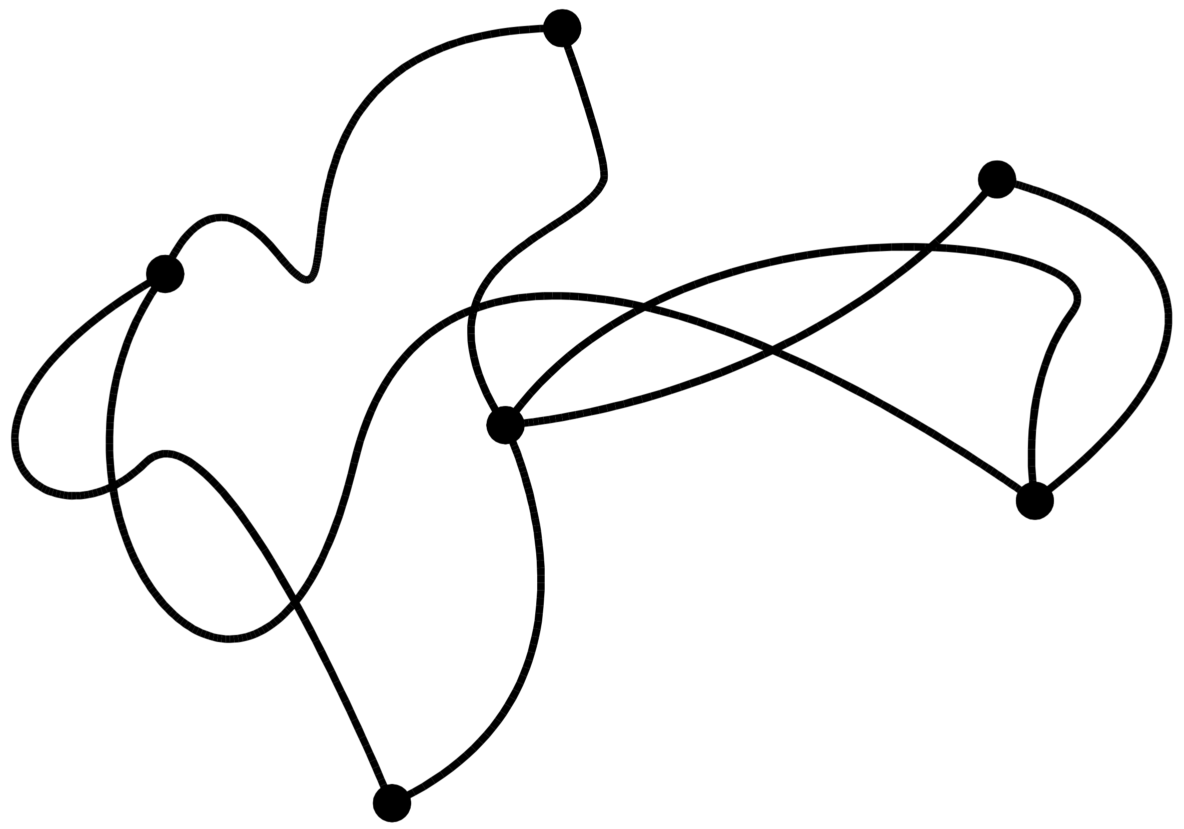
Christian Knauer

Étienne Schramm

Andreas Spillner

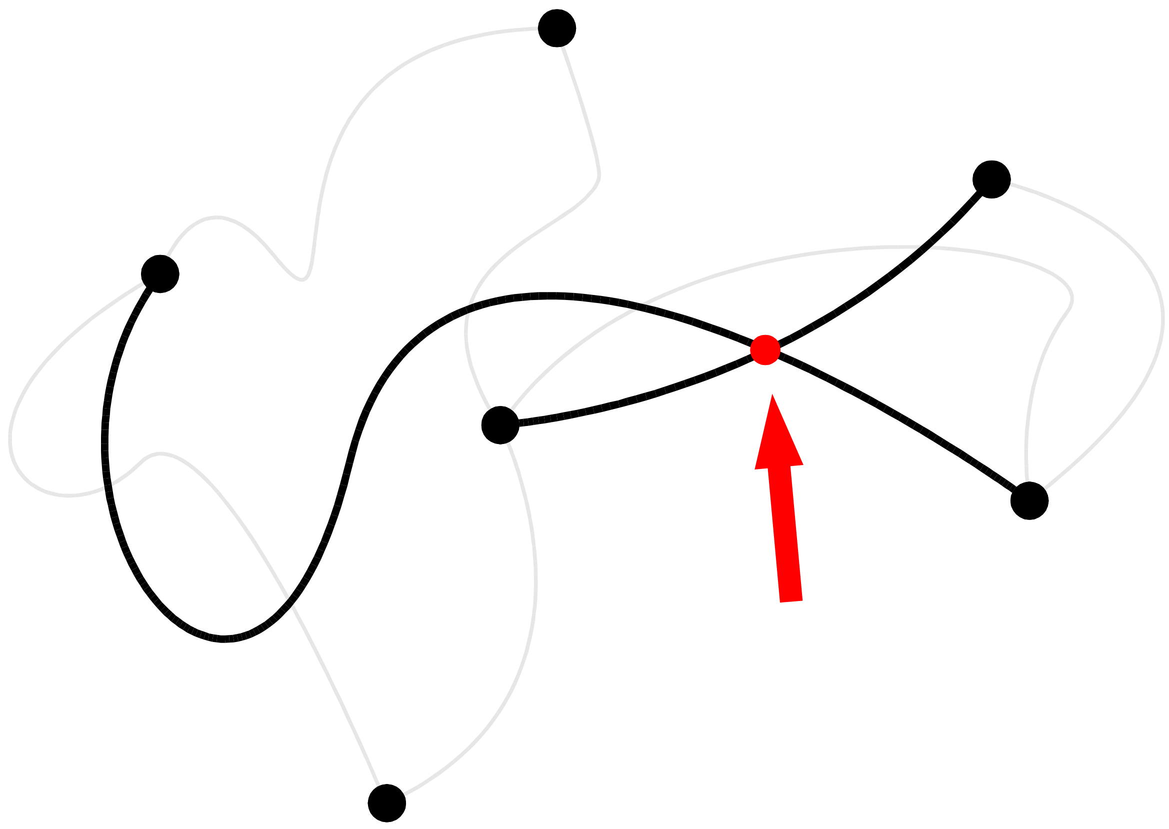
Alexander Wolff

Topological graphs



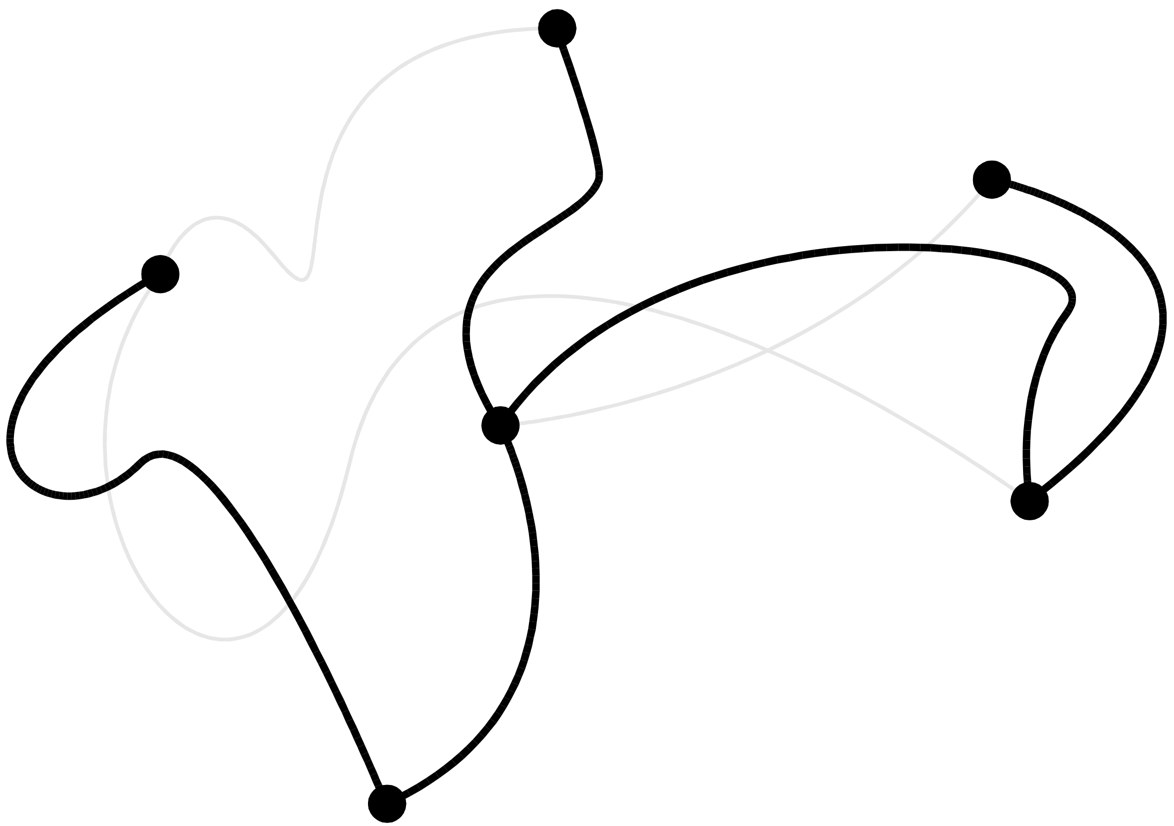
Graph G with a fixed embedding.

Crossings



Pair of crossing edges.

Crossing-free configurations



Known results

Decision problems: Is there a crossing-free spanning tree?

NP-hard

- for general topological graphs

  [Kratochvíl, Lubiw, Nešetřil 1991]

- for very restricted classes of topological graphs

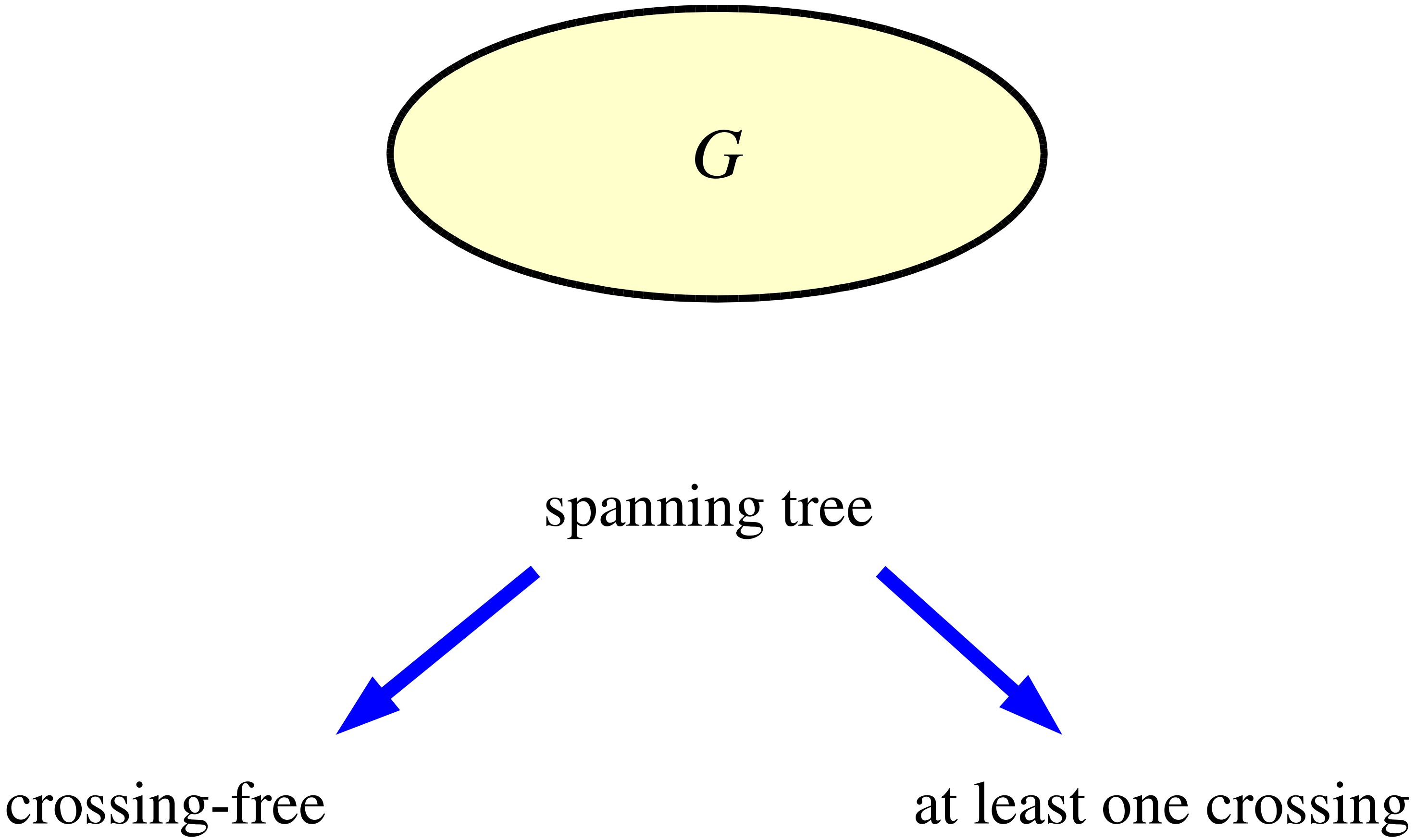
  [Jansen, Woeginger 1993]

Our results

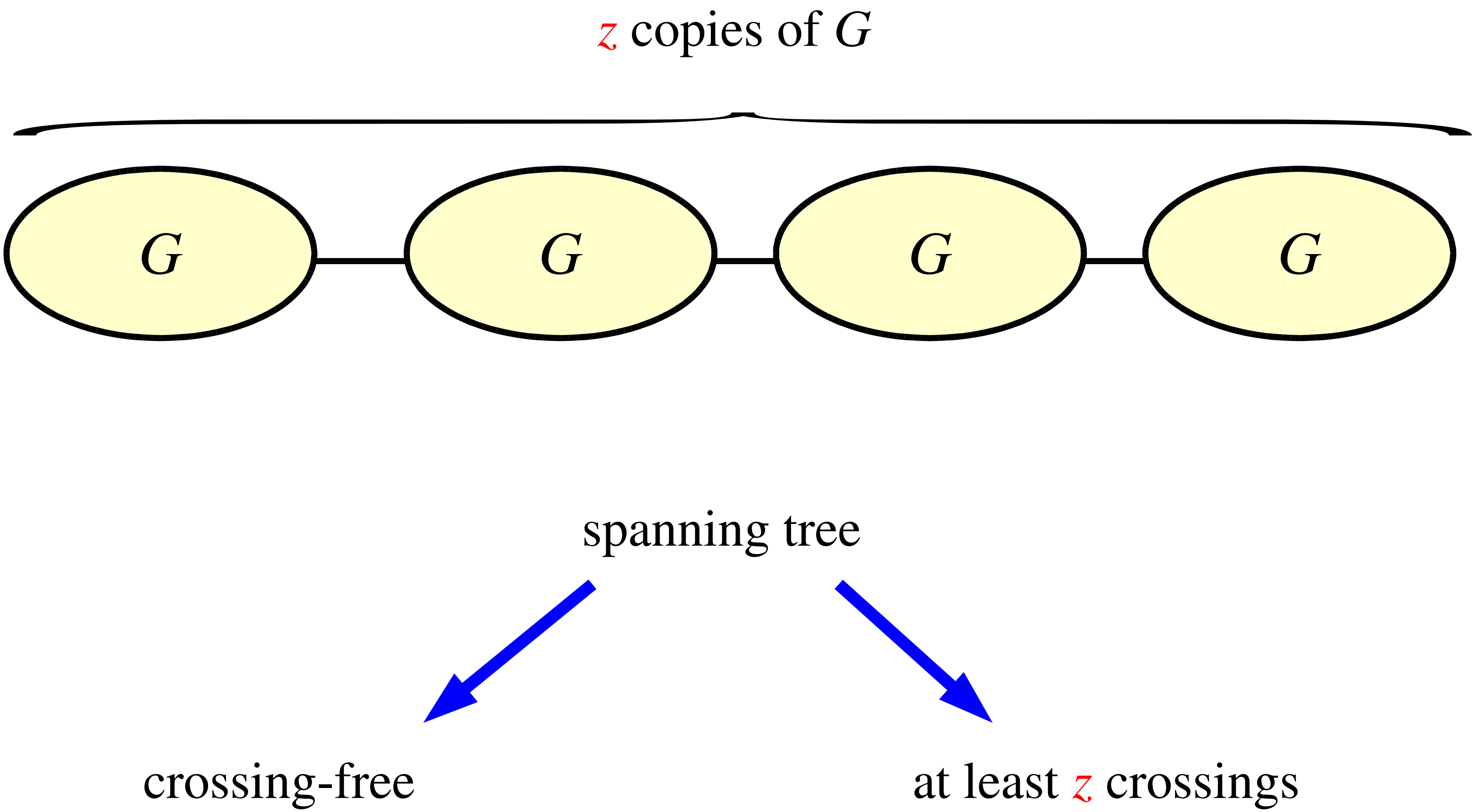
Optimization problems: Find a spanning tree with as few crossings as possible.

- heuristics
- hardness of approximation
- fixed-parameter algorithms
- mixed-integer linear program formulation

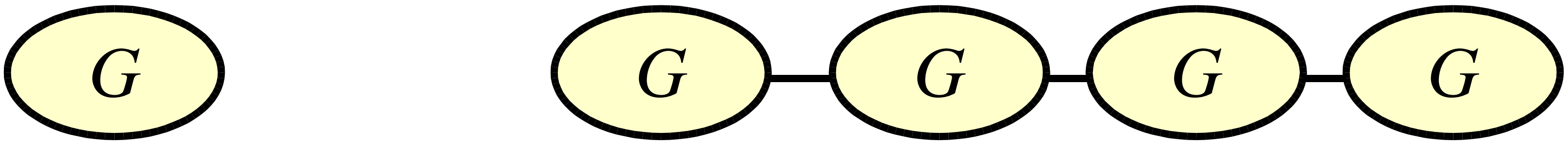
Hardness of approximation



Hardness of approximation



Hardness of approximation



k crossings → $z \cdot k$ crossings

↓

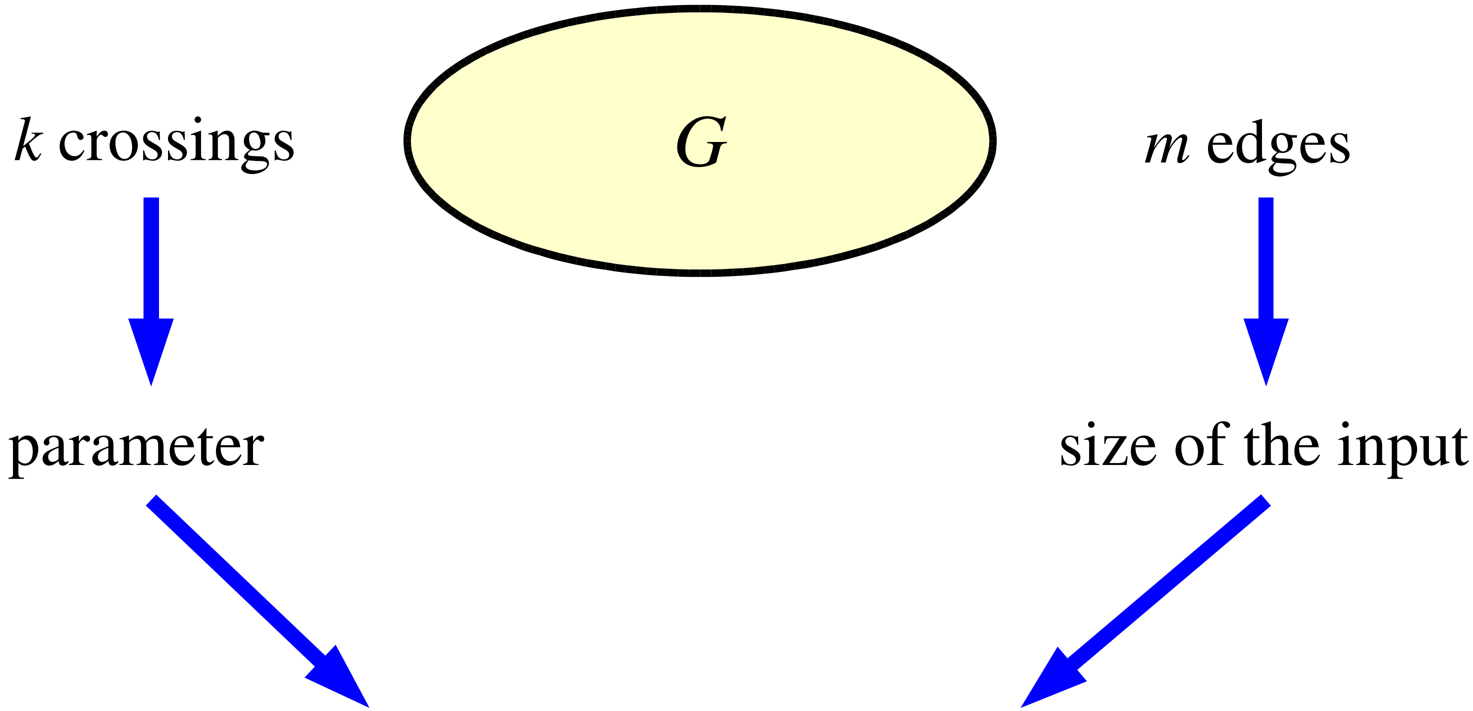
choose $z = k^{\frac{1-\epsilon}{\epsilon}}$

<u>Result</u>

It is NP-hard to approximate the minimum number of crossings in a spanning tree of G within a factor of $k^{1-\epsilon}$ for any $\epsilon > 0$.

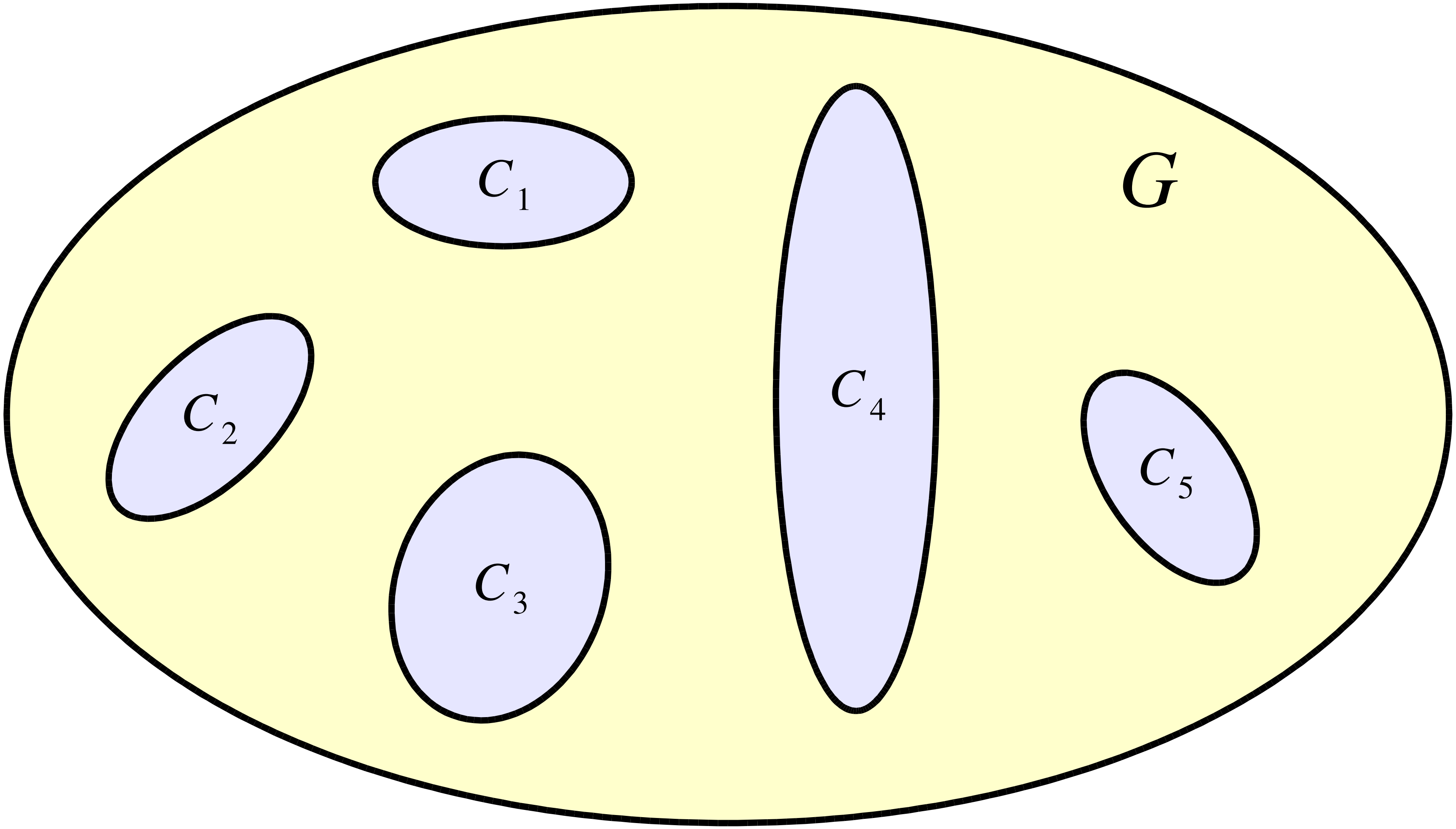
Fixed-parameter algorithms

What makes the problem computationally hard?



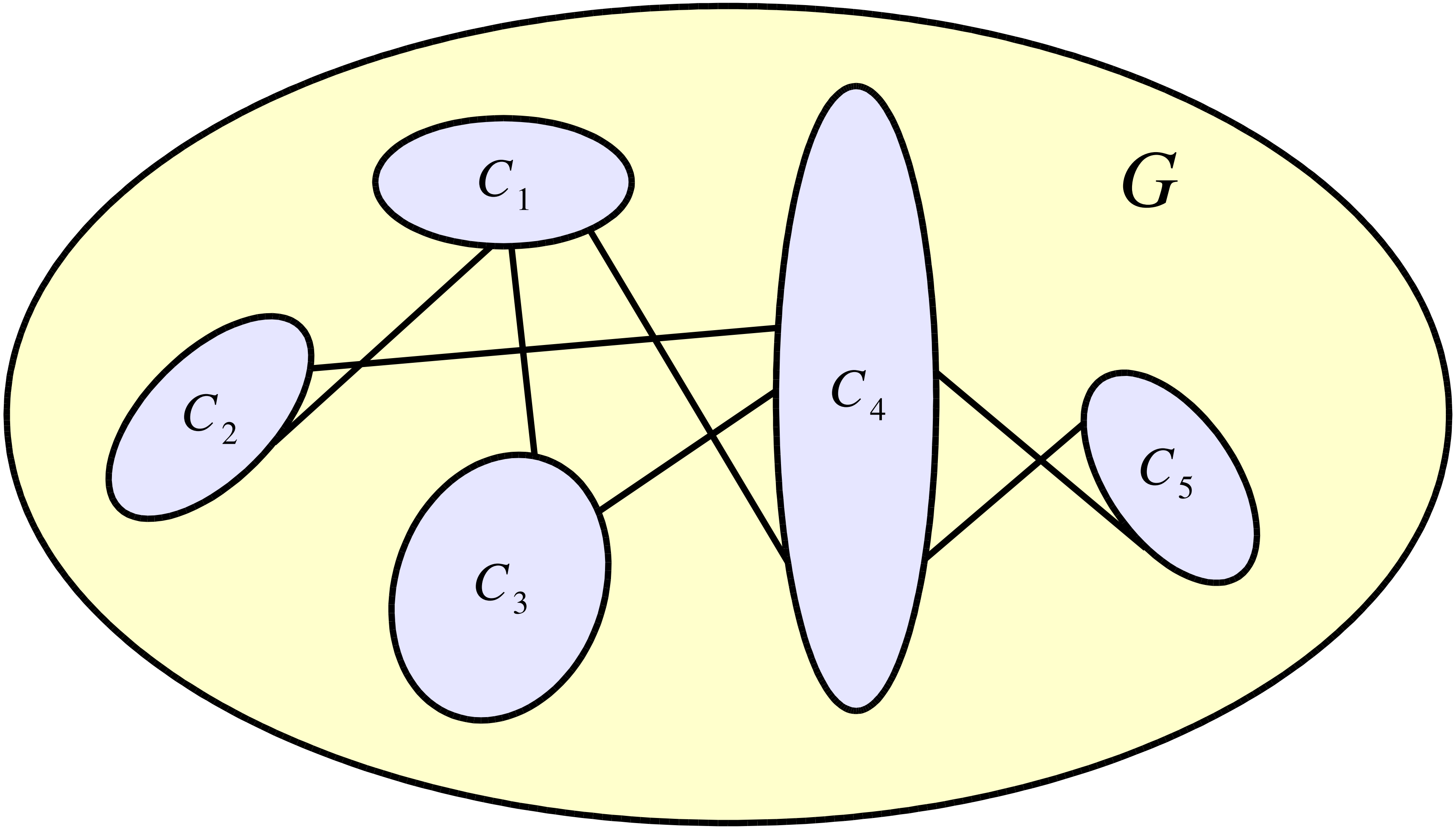
$f(k) \cdot poly(m)$-time exact algorithm

Basic idea



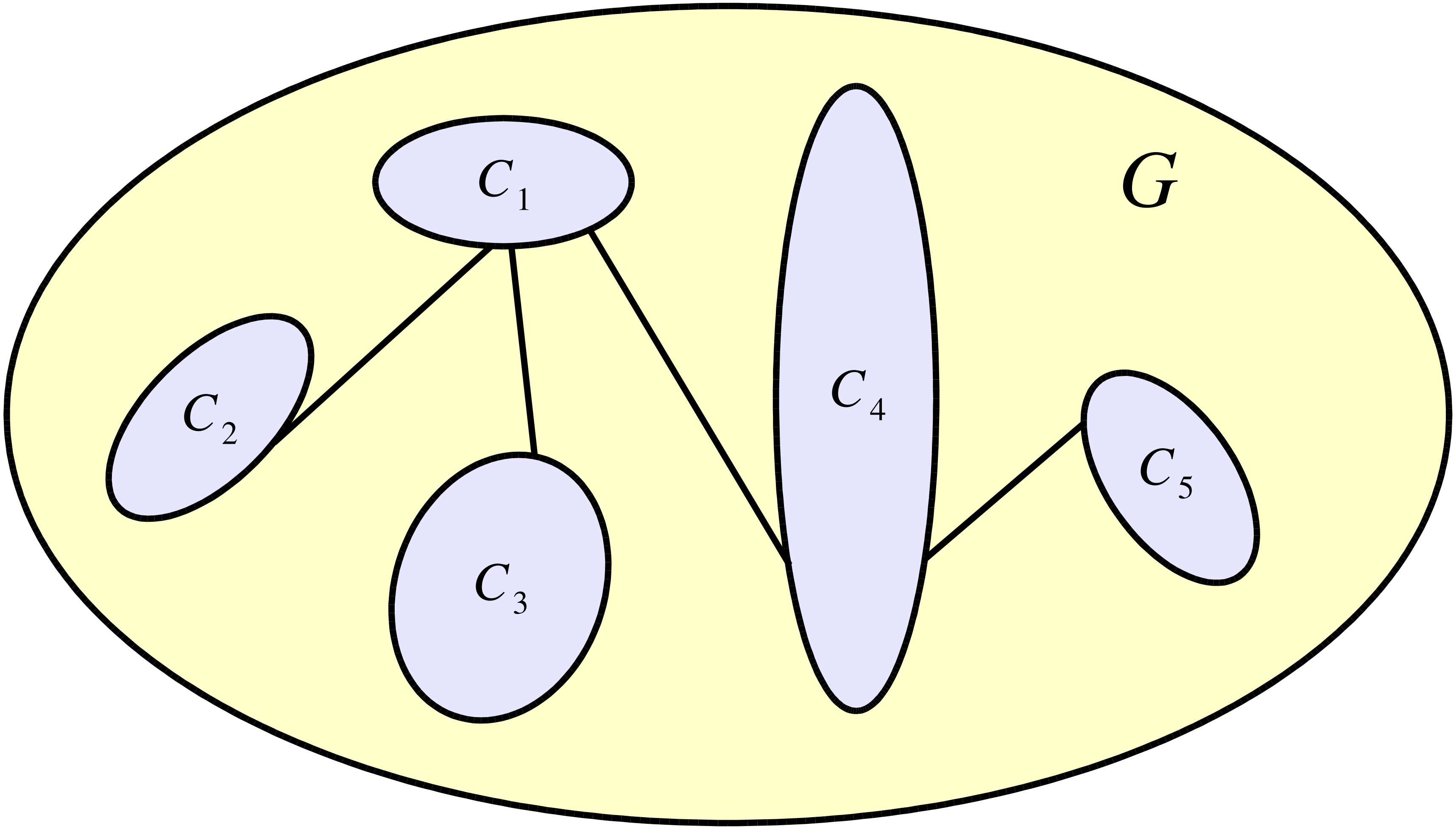
Use edges not participating in any crossing.

Basic idea



At most $2k$ edges remain. → problem kernel

Basic idea



Check every subset of the remaining edges.

Basic idea

After polynomial-time preprocessing:

$\leq 2^{2k}$ subsets to check

↓

$4^k \cdot poly(k)$-time algorithm

Result

The optimization problem is fixed-parameter tractable and can always be reduced to a problem kernel of size at most $2k$.

Simple observations

Observation 1: From every pair of crossing edges at most one can appear in a crossing-free spanning tree.

↓

$\leq 2^k$ subsets to check

↓

$2^k \cdot poly(k)$-time algorithm

Simple observations

Observation 2:

$c^k \cdot poly(k)-$time algorithm for the decision problem

↓

optimal solution in $\sum_{j=0}^{k} \binom{k}{j} c^{k-j} \cdot poly(k) = (c+1)^k \cdot poly(k)$ time

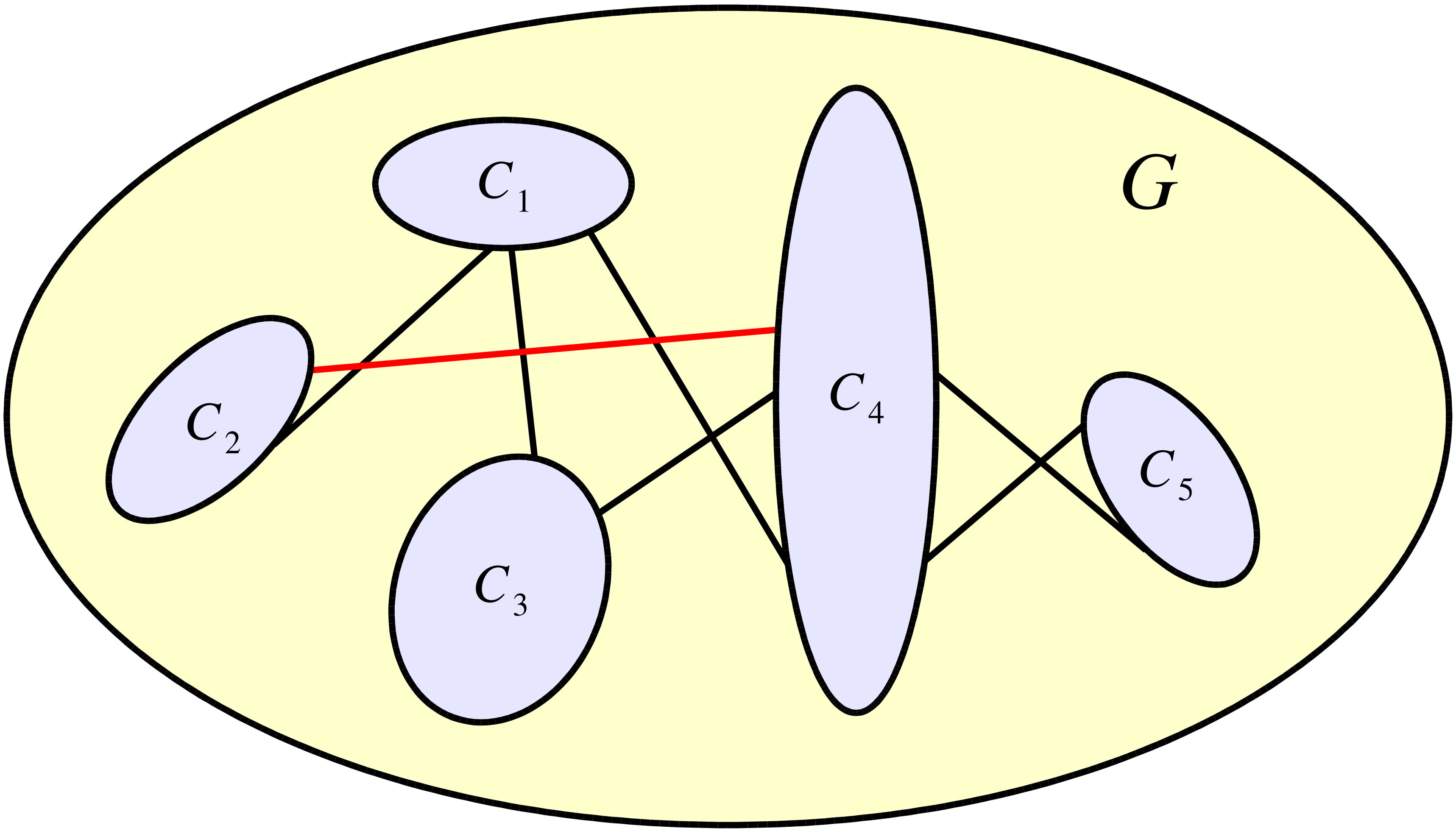
Speeding up

Goal: $c^k \cdot poly(k)$-time algorithm with $c < 2$

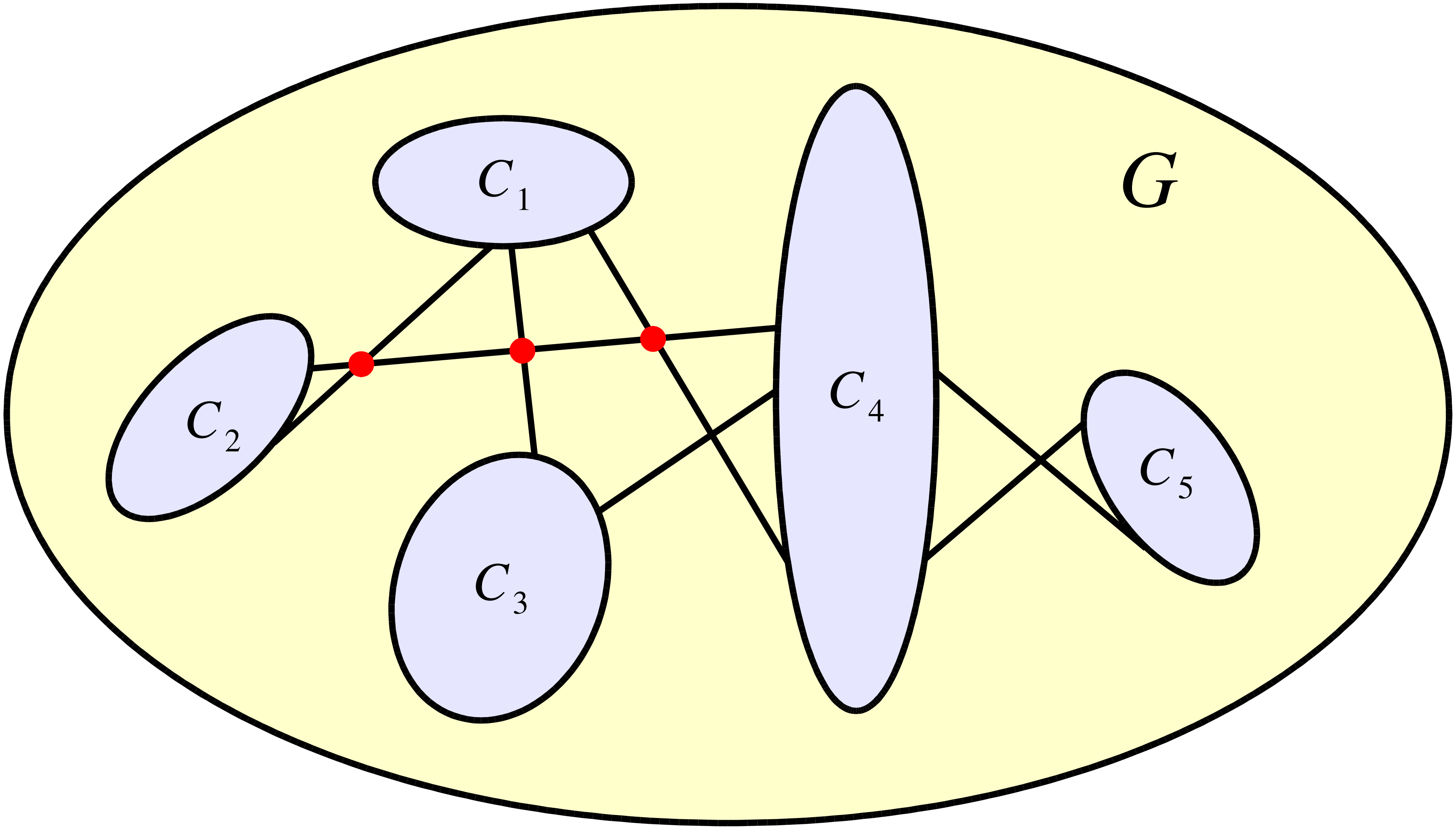
Speeding up

Result: $1.9999996^k \cdot poly(k)$-time algorithm

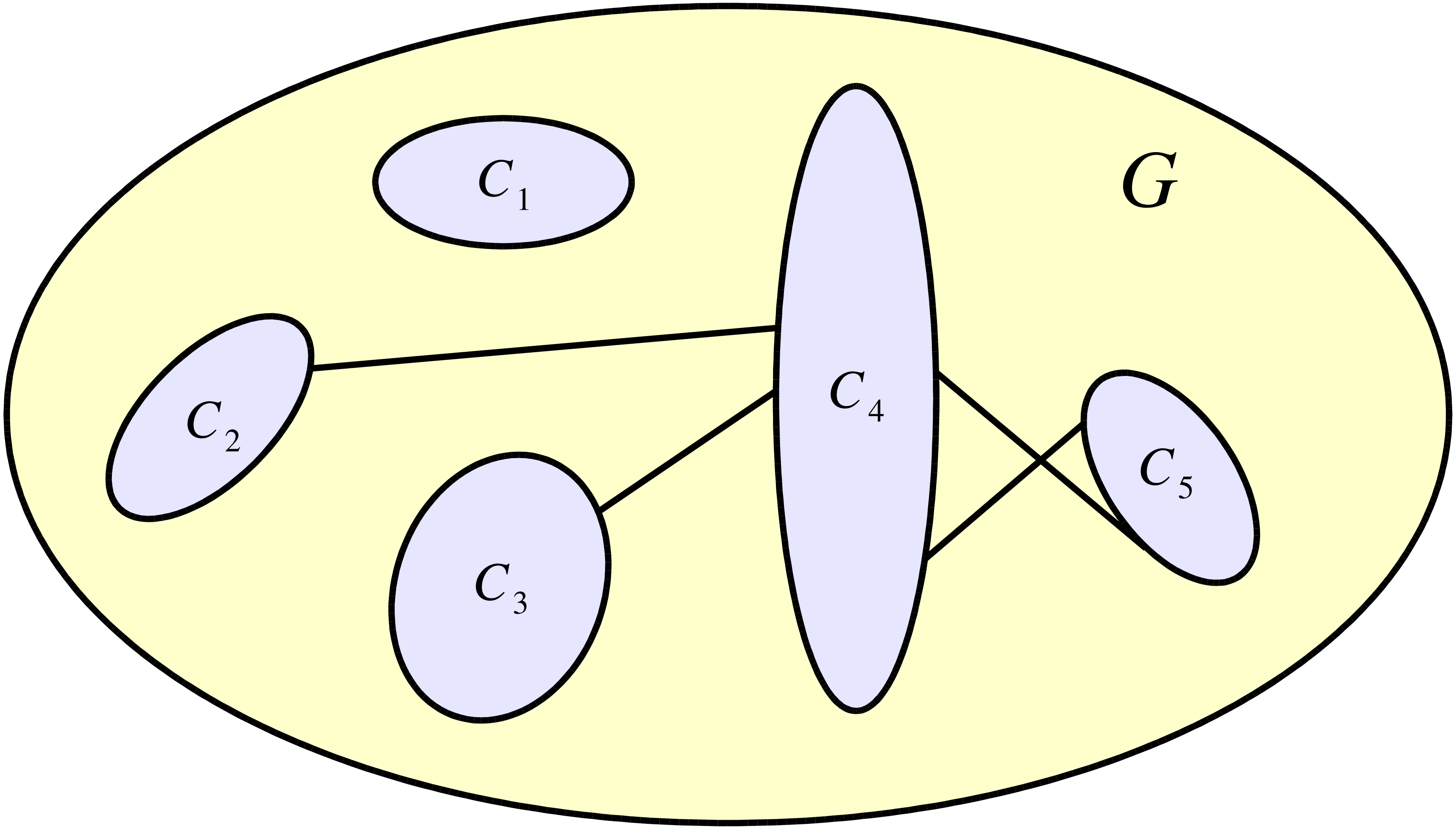
Nice edges



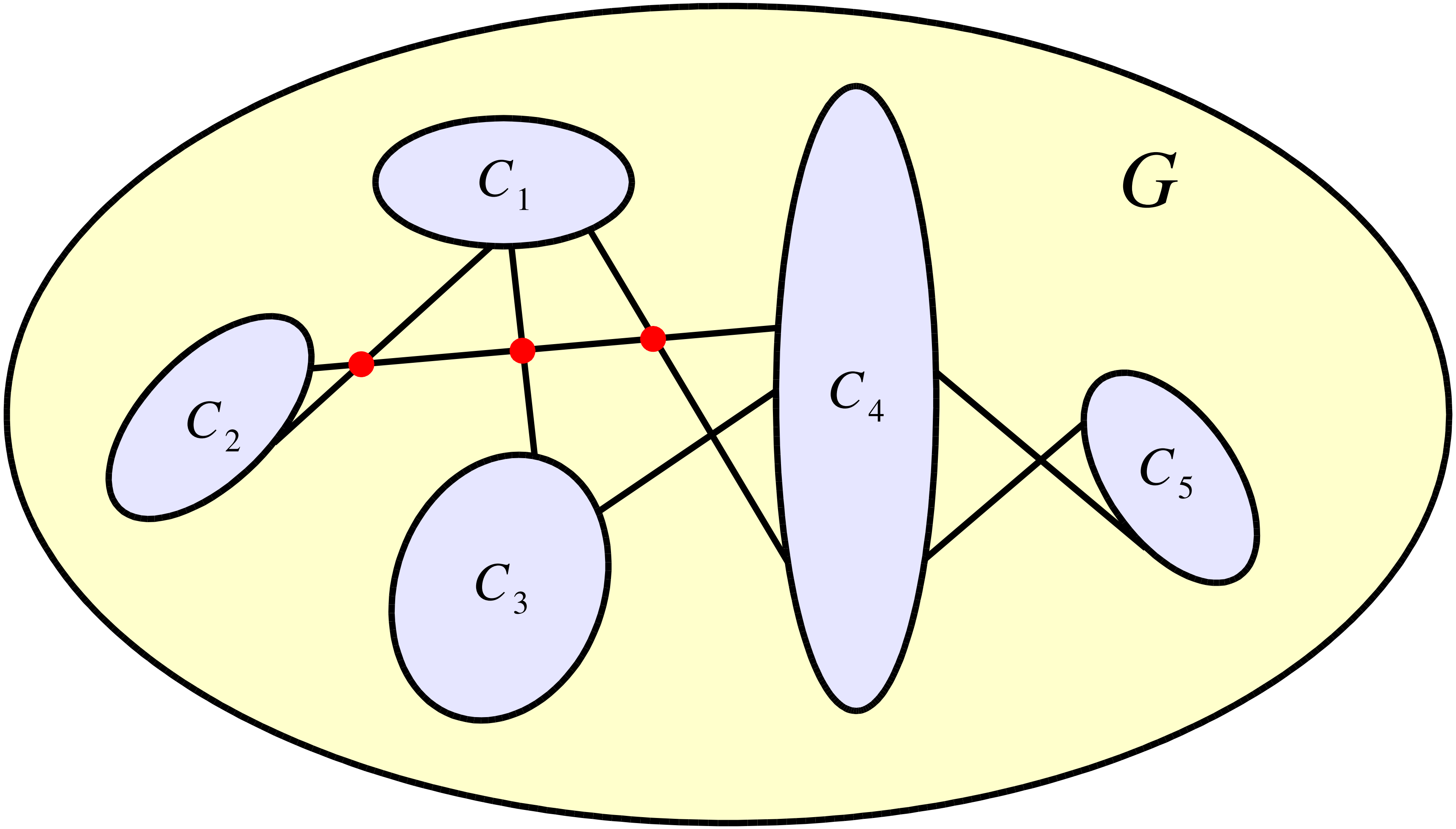
Nice edges



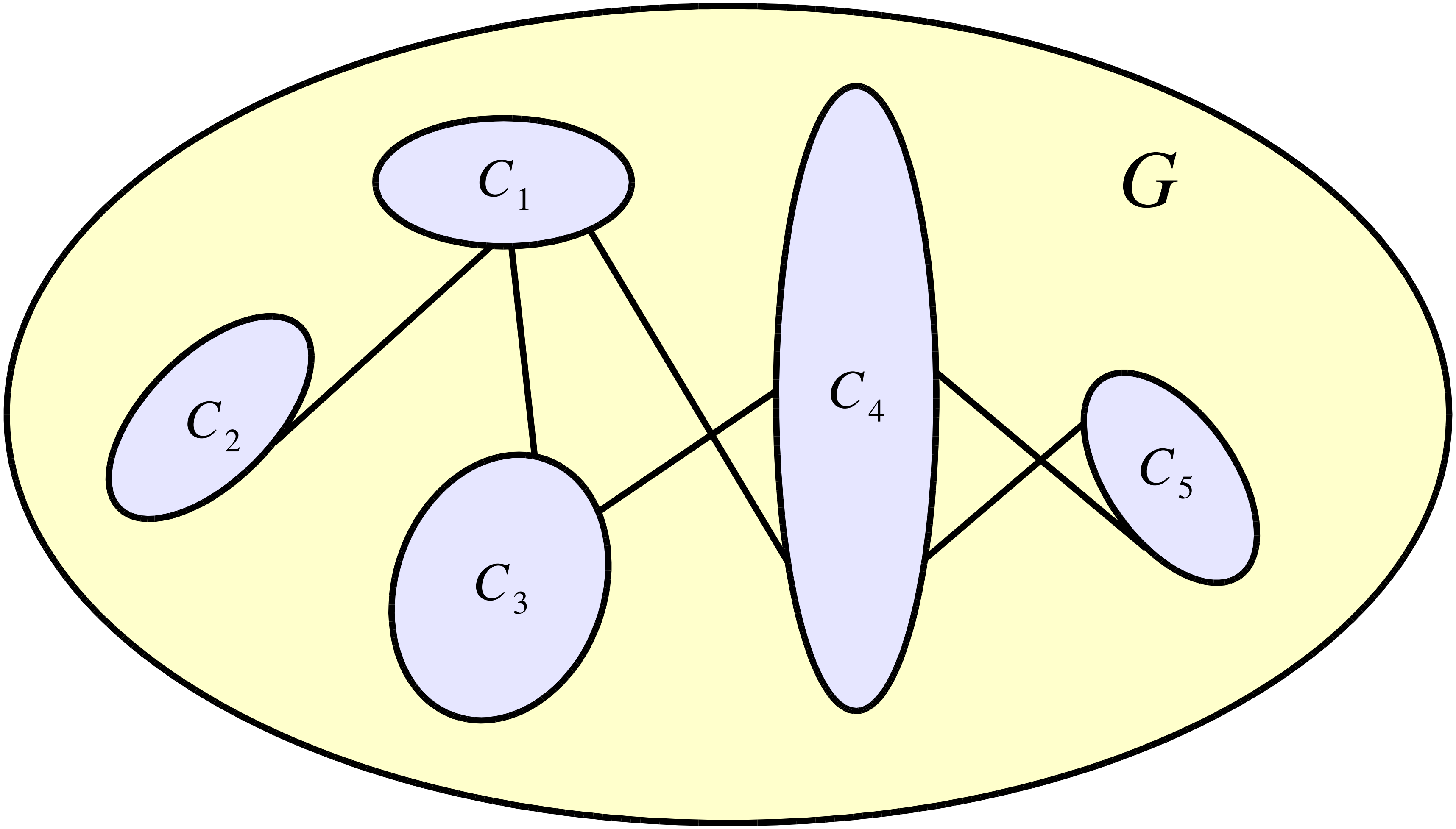
Nice edges



Nice edges



Nice edges



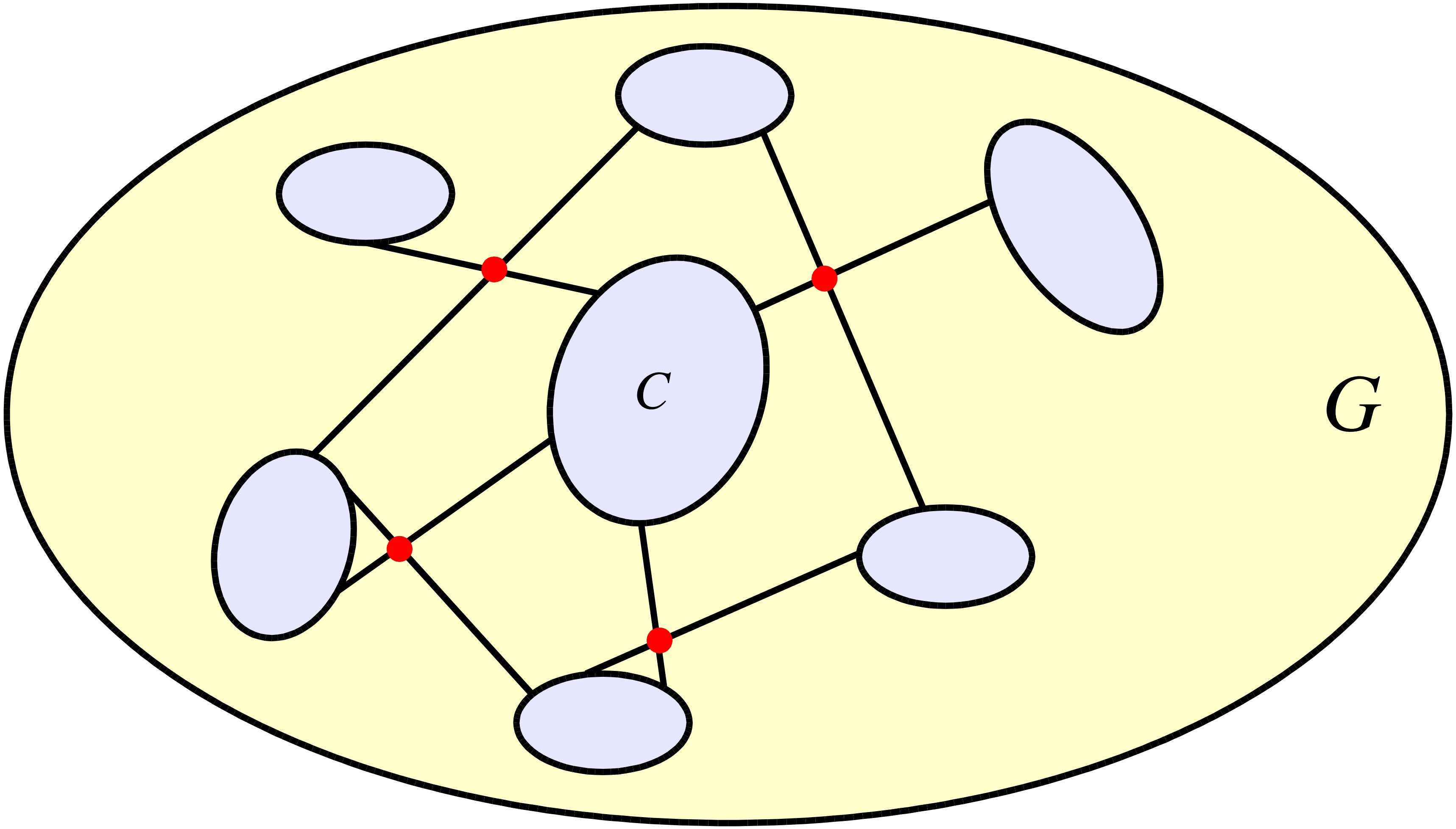
Nice edges

Recurrence: $T(k) \leq 2T(k-z)$ with $z \geq 2$

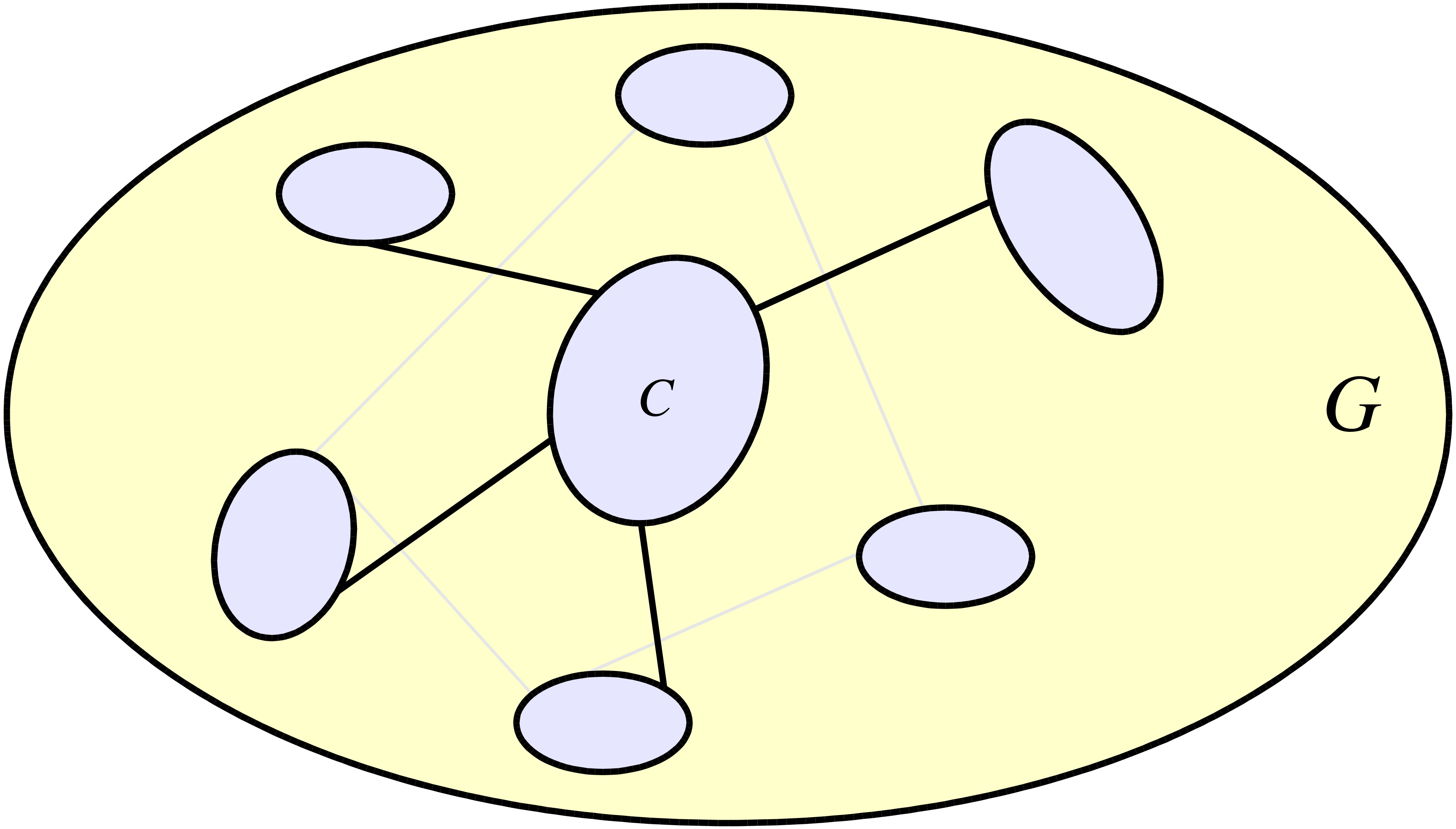
$\Downarrow$

$\sqrt{2}^k \cdot poly(k)$-time algorithm

Disjoint crossings

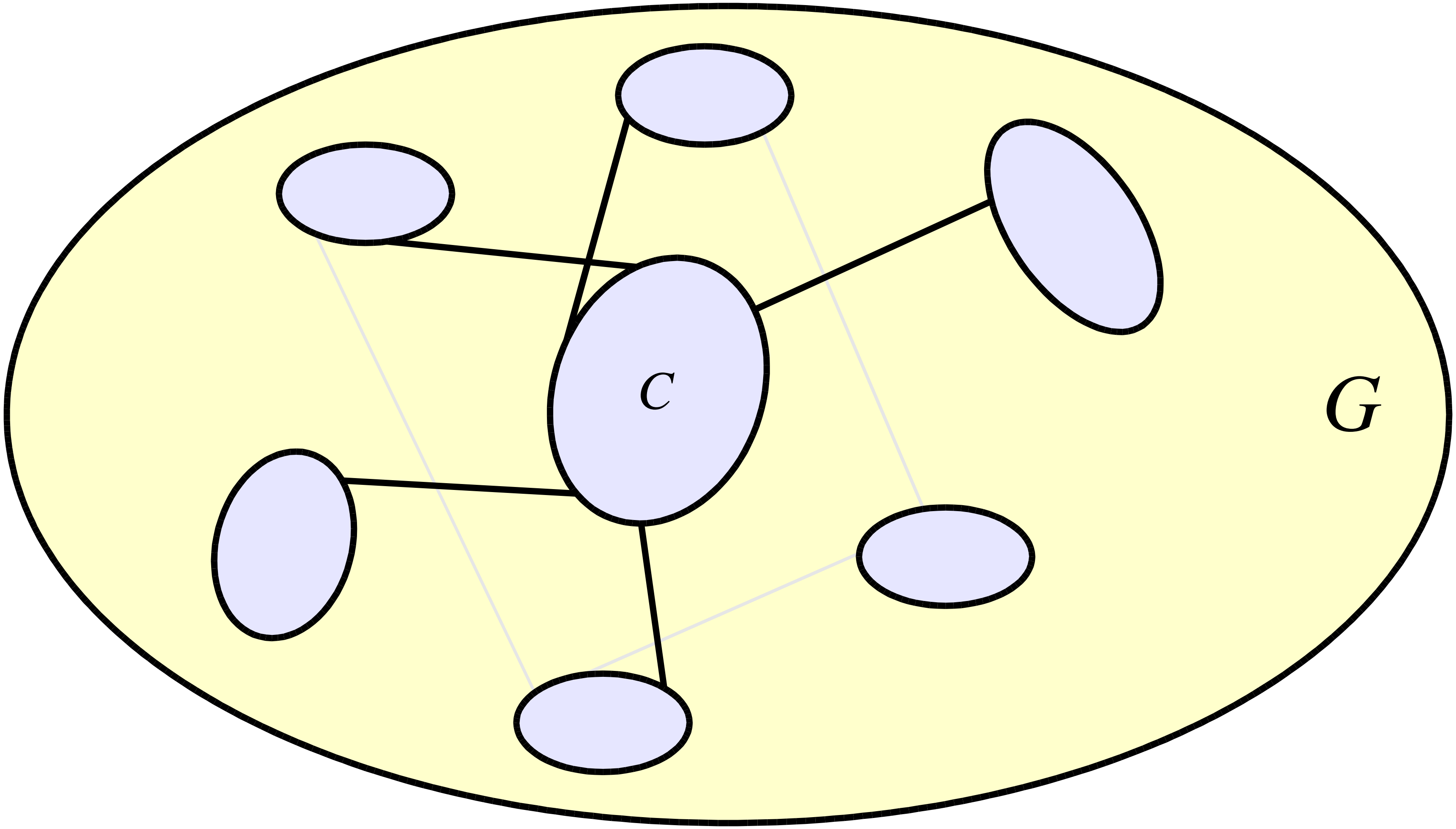


Disjoint crossings



A little bit tricky!

Disjoint crossings



Disjoint crossings

Case 1:

Many components → Component with few incident edges.

Case 2:

Few components → Few edges suffice to build a spanning tree.

Future Work

- approximation factor in $o(k)$
- faster fixed-parameter algorithms:

$$c^k \cdot poly(k)$$

- other parameters
- further similar problems
- implementation and evaluation

Thank You