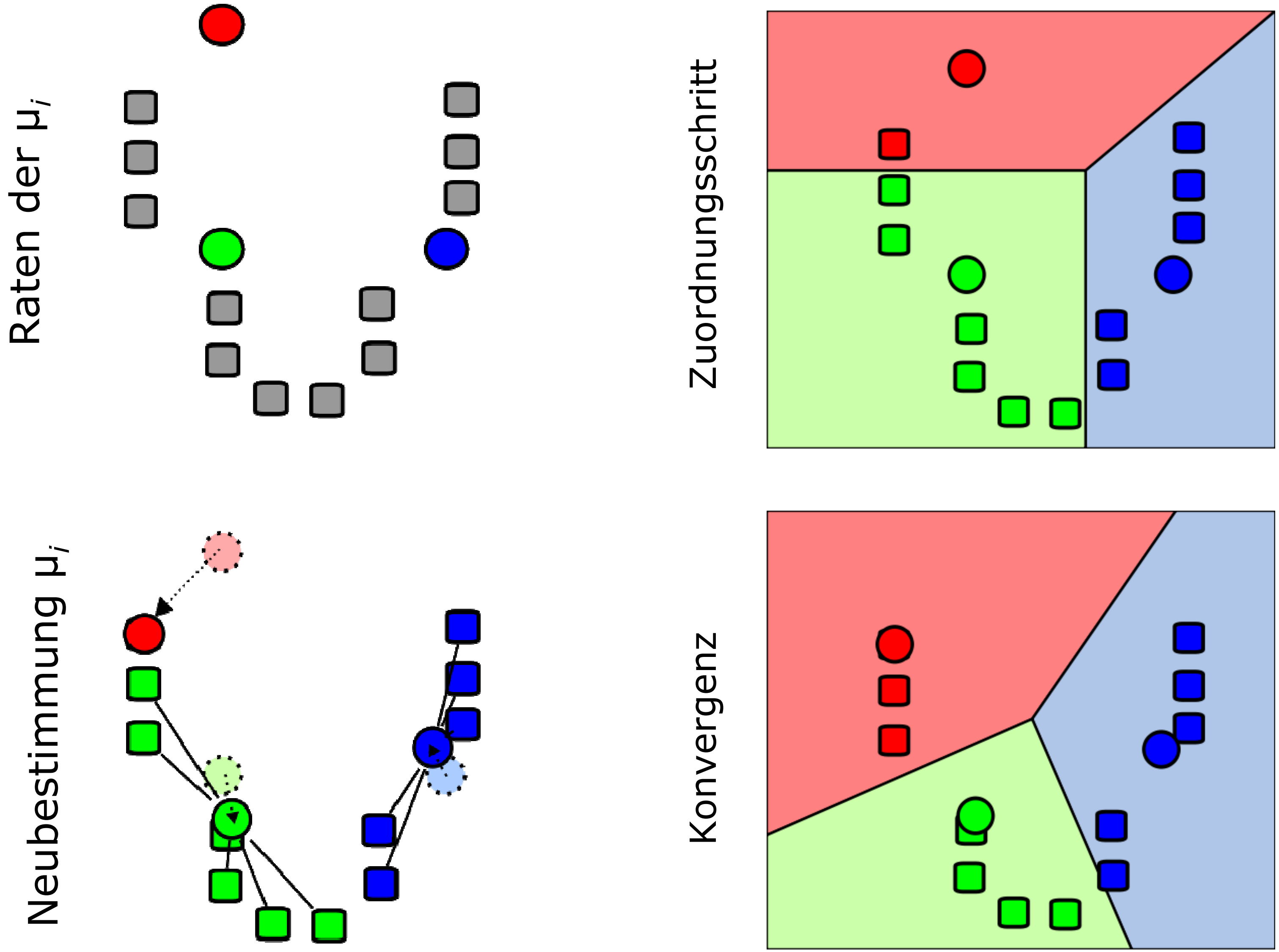
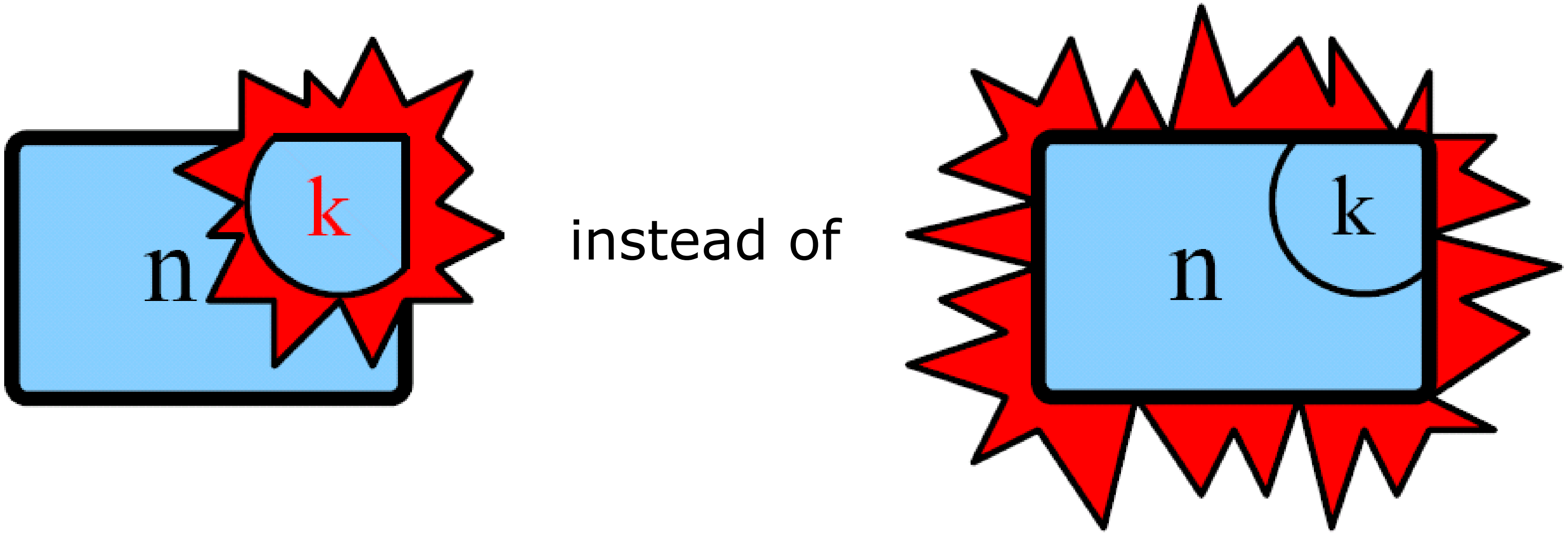
k-means Clustering



Parameterized algorithms and FPT

- we want to find exact solutions for NP-hard problems (approximations are no good for bioinformatics)
- problem size *n*, parameter *k* (much smaller than *n*)

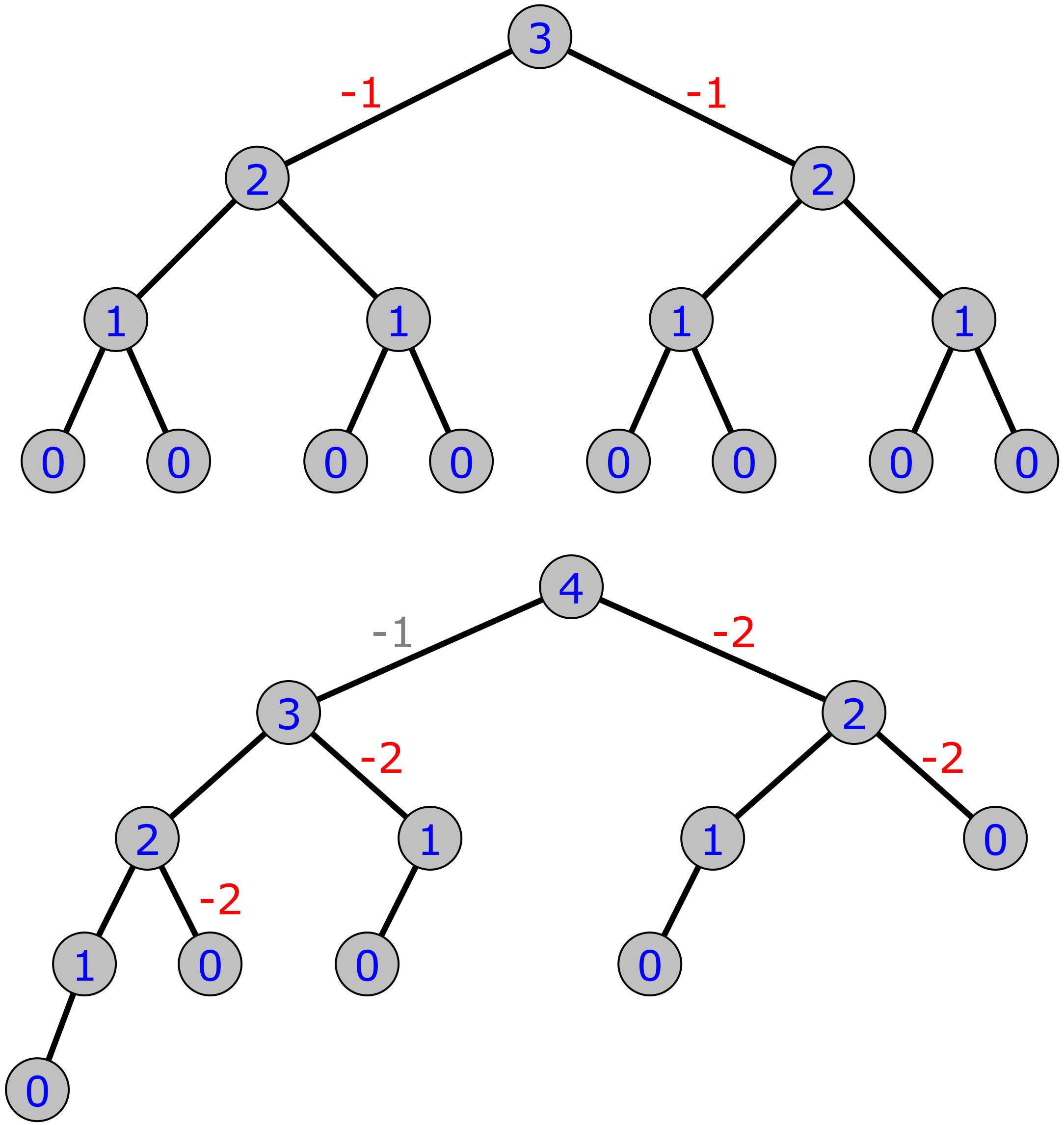


- idea: accept combinatorial explosion, but limit it to a rather small parameter *k*

Preliminaries: Parameter *k*

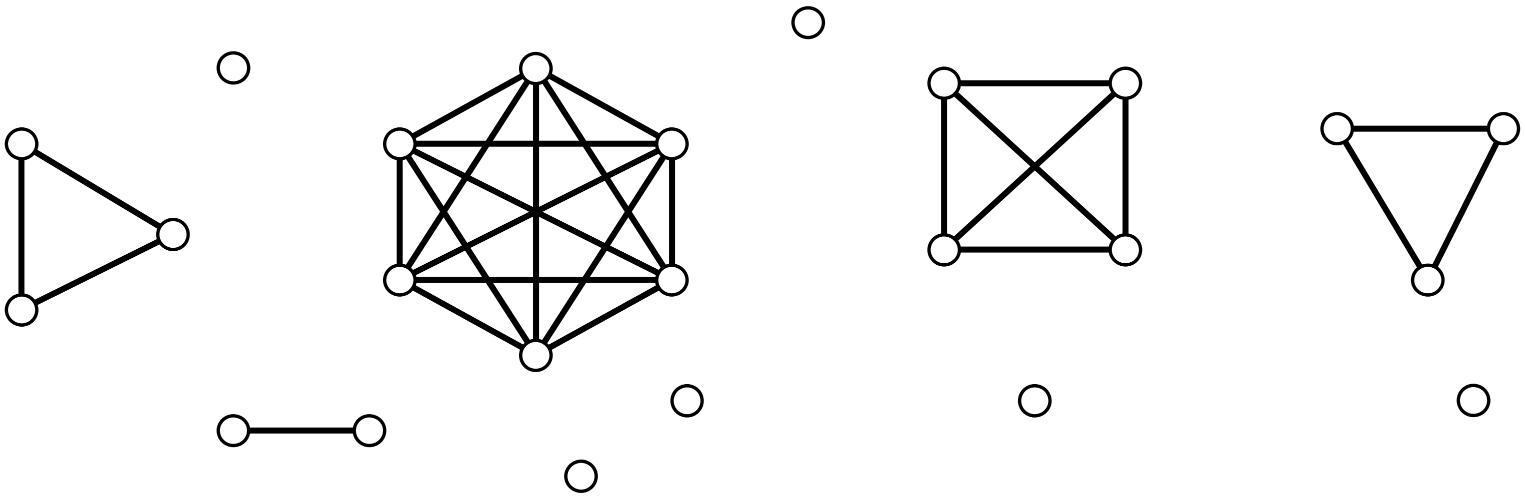
- parameter *k* is certain feature of the instance (size of the solution, deviation from triviality, …)
- we often assume that parameter *k* is given
- answers of our algorithm: either "no solution" or a solution with parameter $k' \leq k$
- this is no major difficulty: call the algorithm repeatedly for $k = 1,2,3,\ldots$ until a solution is found
- running time (super-)exponential in *k*, so the last program call dominates the running time

Preliminaries: Bounded search trees



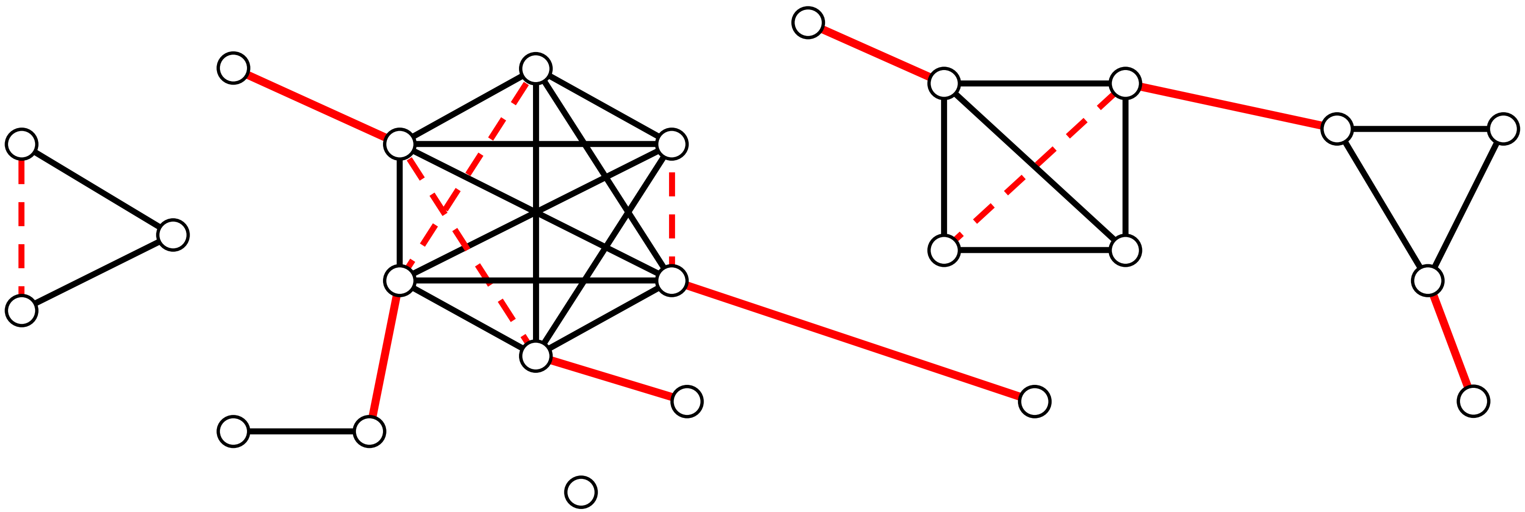
- complete binary tree of height k has 2^k leaves
- equival. formulation: we reduce our initial parameter k to zero
- what if we reduce k by other values than one?
- branching vector (1,2) with branching number 1.62
- tree size $O(1.62^k)$

Cluster Editing Problem



- represent objects as vertices of an undirected graph
- connect similar objects by an edge
- cluster vertices into disjoint union of cliques
- unfortunately, our input data contains errors…

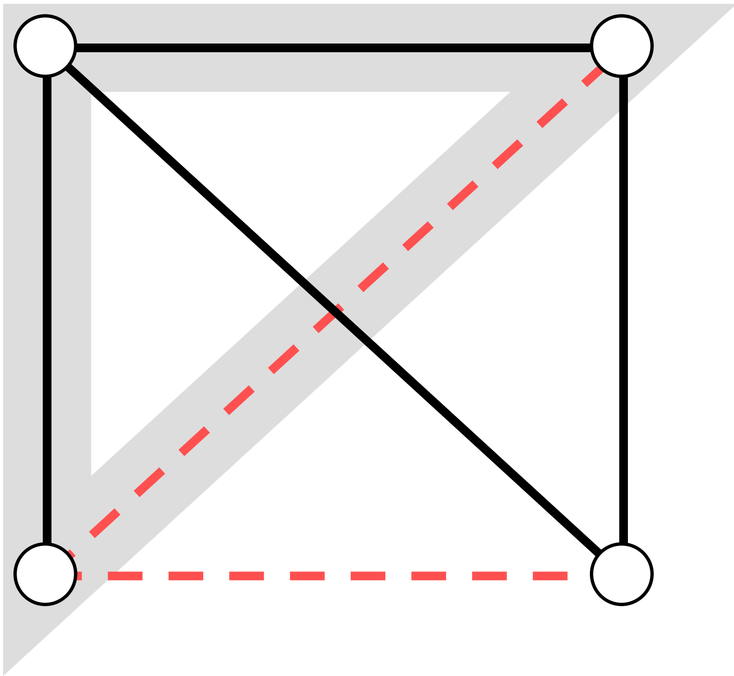
Cluster Editing Problem



- our task: find smallest set of edge modifications so that graph becomes disjoint union of cliques
- NP complete problem, also APX hard
- efficient parameterized algorithms for

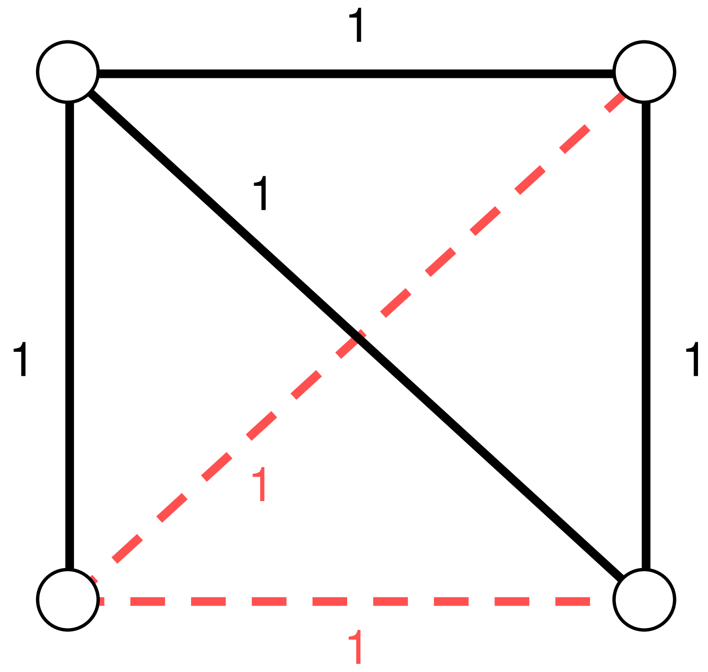
  k = number of edge modifications
- fastest known algorithm running time $O(1.92^k + n^3)$
- fastest implemented alg. running time $O(2.27^k + n^3)$

Conflict triples



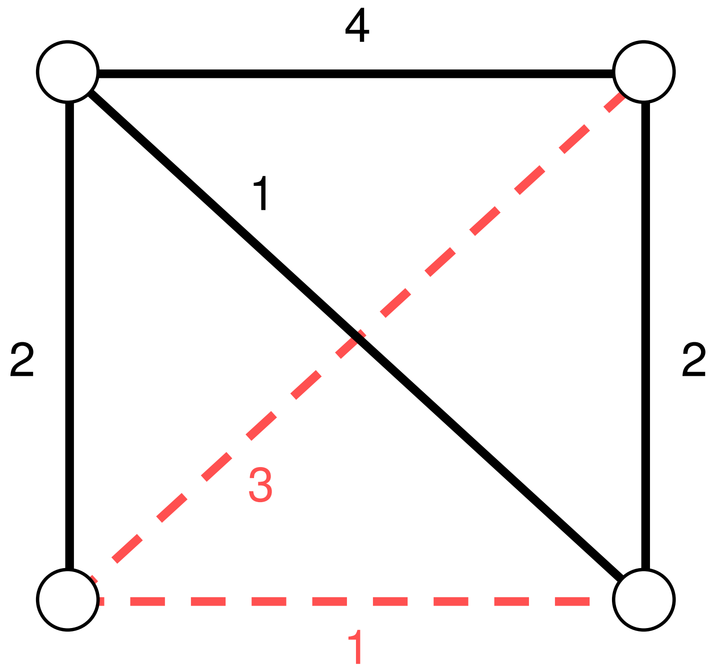
- a conflict triple is an induced path of length two
- easy to see: a graph is a disjoint union of cliques iff it contains no conflict triples
- FPT idea: find a conflict triple, branch to resolve it

Going weighted



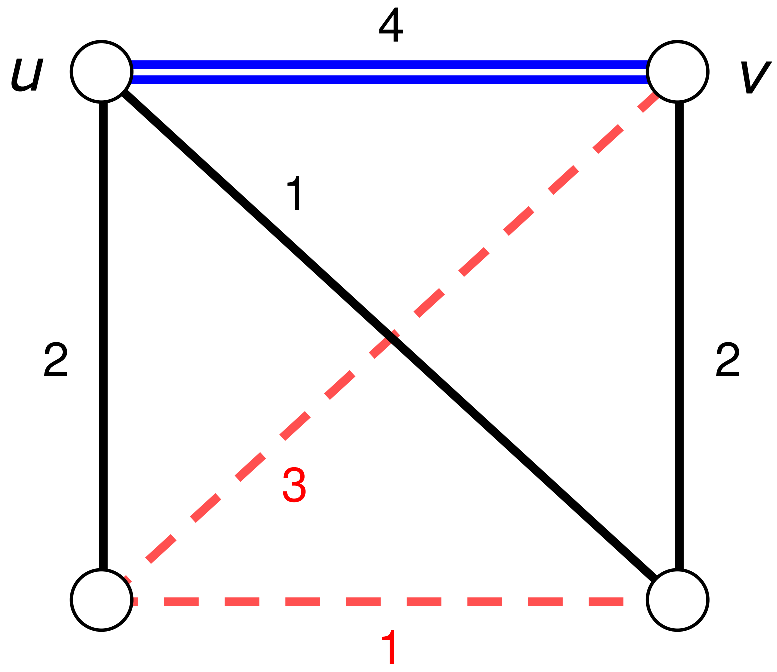
- we give every edge and non-edge an insertion or deletion cost
- transform unweighted instance into weighted instance by assigning cost one to all edges and non-edges

Going weighted



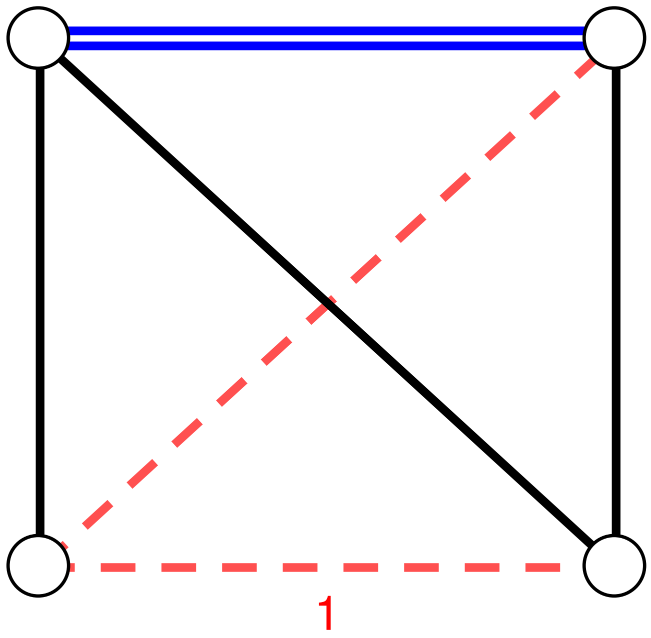
- we assume that all edges and non-edge have arbitrary integer costs
- find set of edge modifications of minimal total weight k
- no edges with cost zero allowed in the input

Merging edges



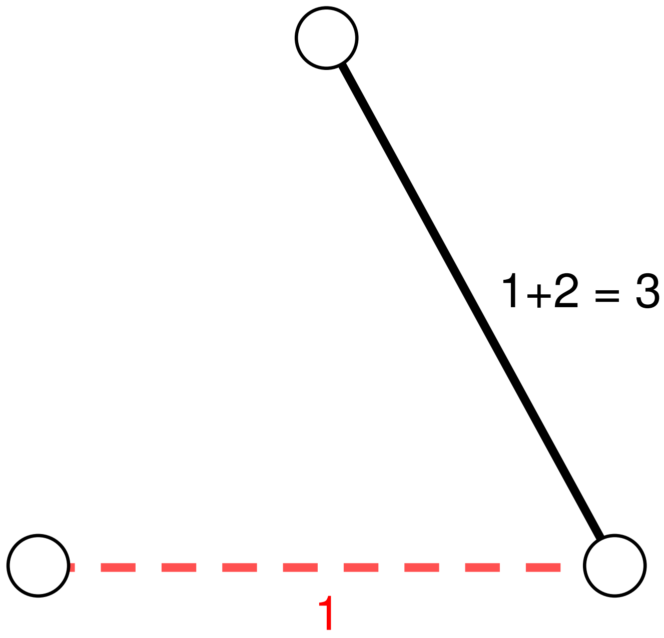
- if we know that a certain edge has to be part of an optimal solution, we can set it to permanent
- in any solution, *uw* is an edge iff *vw* is an edge

Merging edges



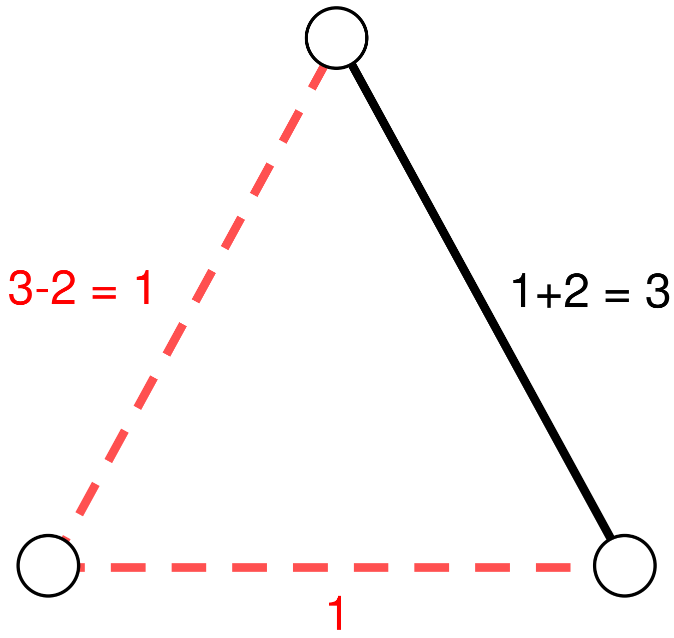
- if we know that a certain edge has to be part of an optimal solution, we can **merge it**

Merging edges



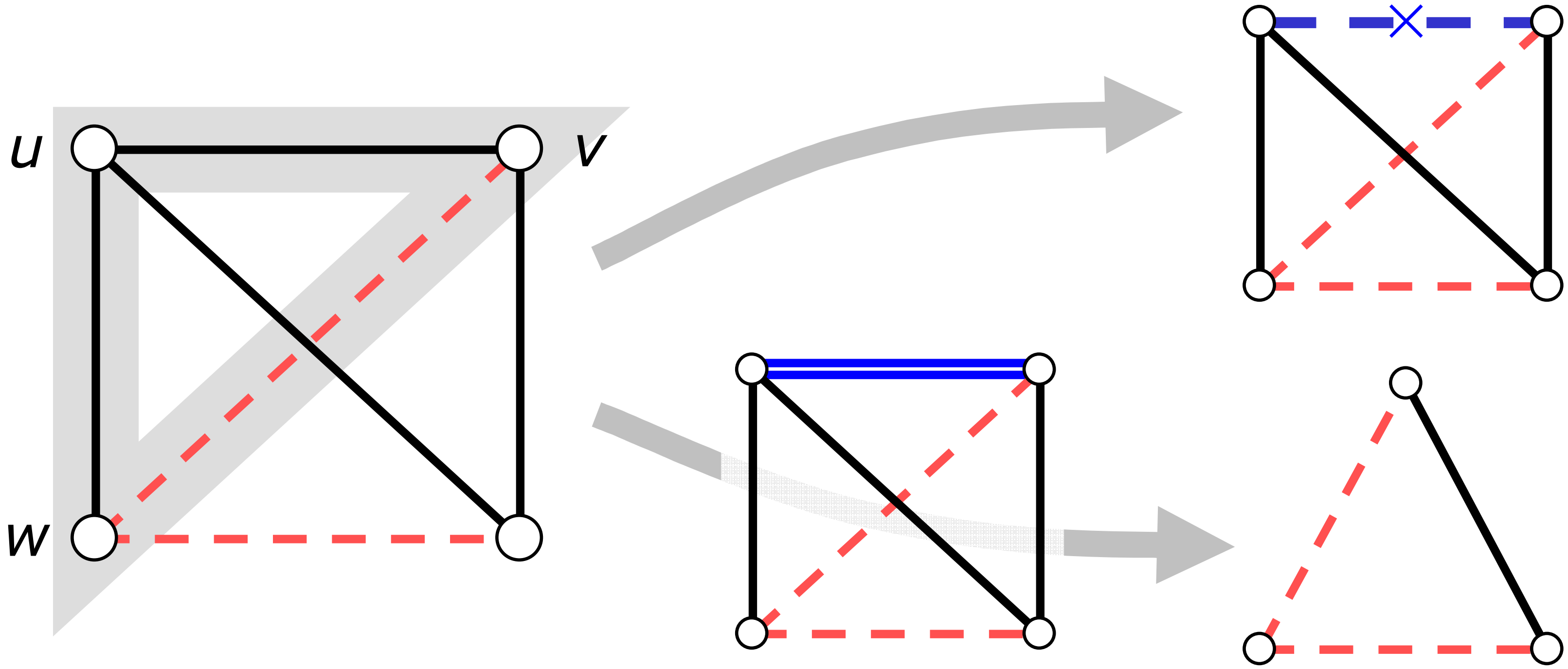
- both edges present or absent: add costs

Merging edges



- both edges present or absent: add costs
- merge edge and non-edge: subtract smaller cost
- immediately decrease k by smaller cost

Edge branching strategy



- let *vuw* be a conflict triple such that *uv* is an edge
- branch into two cases:
  1. delete *uv* and set it to forbidden
  2. set *uv* to permanent and merge it

Running time analysis

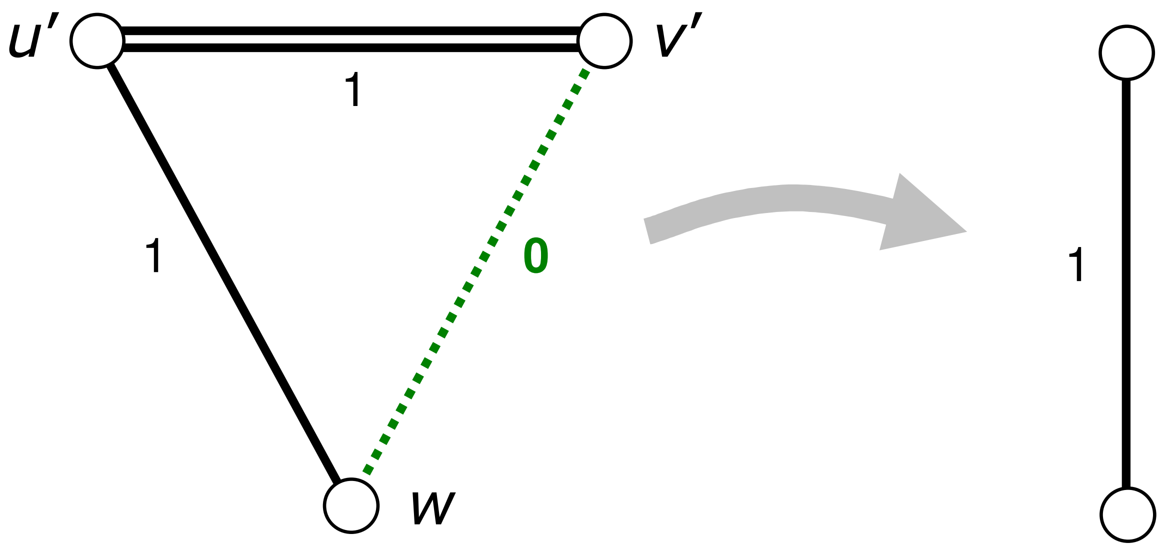
- we want to show that in both cases of edge branching, we decrease parameter k by at least one
- then, the search tree has size $O(2^k)$

- removing edge *uv* generates cost one
- merging *uv* either removes an edge *vw* or inserts a non-edge *vw*, whatever has smaller costs assigned
- in both cases, we generate cost one
- ….so, the branching vector is (1,1) and we are done!

WRONG!

Zero edges

- merging edge *uv* may generate an edge of zero weight, if costs of *uw* and *vw* are equal
- for finding conflict triples, we count zero-edges as being non-existing
- but at a later stage, merging this zero-edge generates no cost!



Bookkeeping

- simple solution: whenever we generate a zero-edge,
  - we only count the merge operation to cost ½, and
  - we "put away" cost ½ for later use
- now, "destroying" a zero-edge also costs ½
- so, edge branching has branching vector (1,½) and the search tree has size $O(2.62^k)$

…can we do better than that?

Edge branching revisited

- unless the weighted instance is a clique, or a clique with a single zero-edge (and, hence, solved)…
- we choose that edge *uv* such that branching on *uv* (delete, merge) leads to minimal branching number
- we can always find an edge that is part of two conflict triples, where zero-edges count as non-existing
- then, branching on *uv* has branching vector (1,2·½) = (1,1) and search tree size $O(2^k)$

We can solve (integer weighted) Cluster Editing in running time $O(2^k + n^3)$.

Refined edge branching

- If there is an edge with branching vector $(1, {}^{3}/_{2})$ or better then we branch on this edge.
- If there is an edge xy and a vertex z in G_w such that x,y,z form a triangle, and if there exist two additional vertices v_1,v_2 such that for both v_1,v_2 one of the following conditions holds (where x and y may be exchanged):
  - xv_i is an edge and yv_i is a non-edge
  - xv_i is a zero-edge and yv_i is a zero-edge
  - xv_i is a zero-edge and yv_i is a non-edge, and zv_i is an edge or a zero-edge
  - xv_i is an edge and yv_i is a zero-edge, and zv_i is a non-edge or a zero-edge

  Then branch on xy.

We can solve (integer weighted) Cluster Editing in running time $O(1.82^k + n^3)$.

Running times, artificial data

number vertices	40	50	60	70	80
average # edit	116	183	263	352	459
2.42^k no merging	31min	5 h		days...	
3^k with merging	29 s	8 min	27 h	58 h*	19 d*
edge branching	3 s	18 s	6 h	18 h	14 h

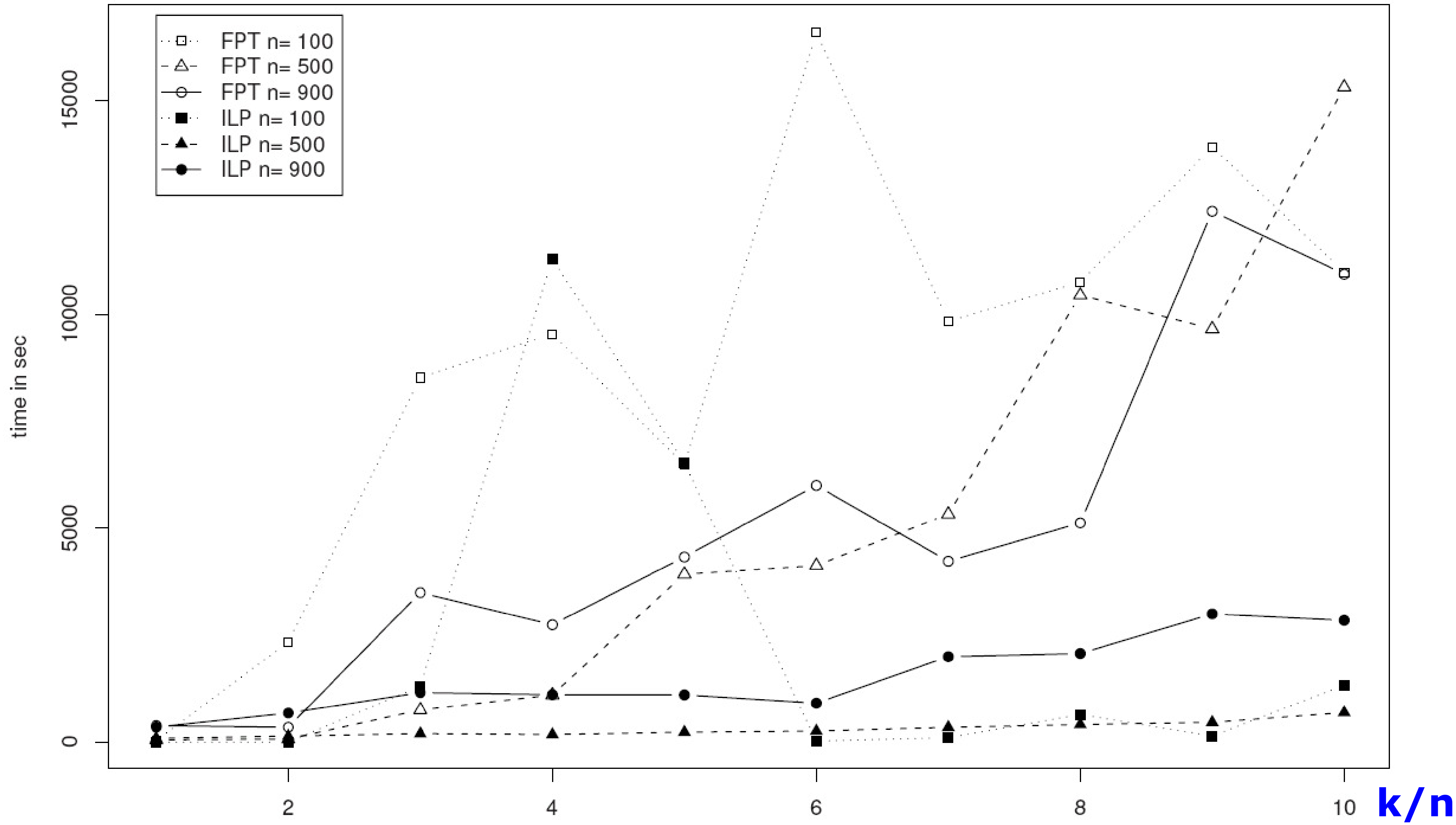
Running times, artificial data

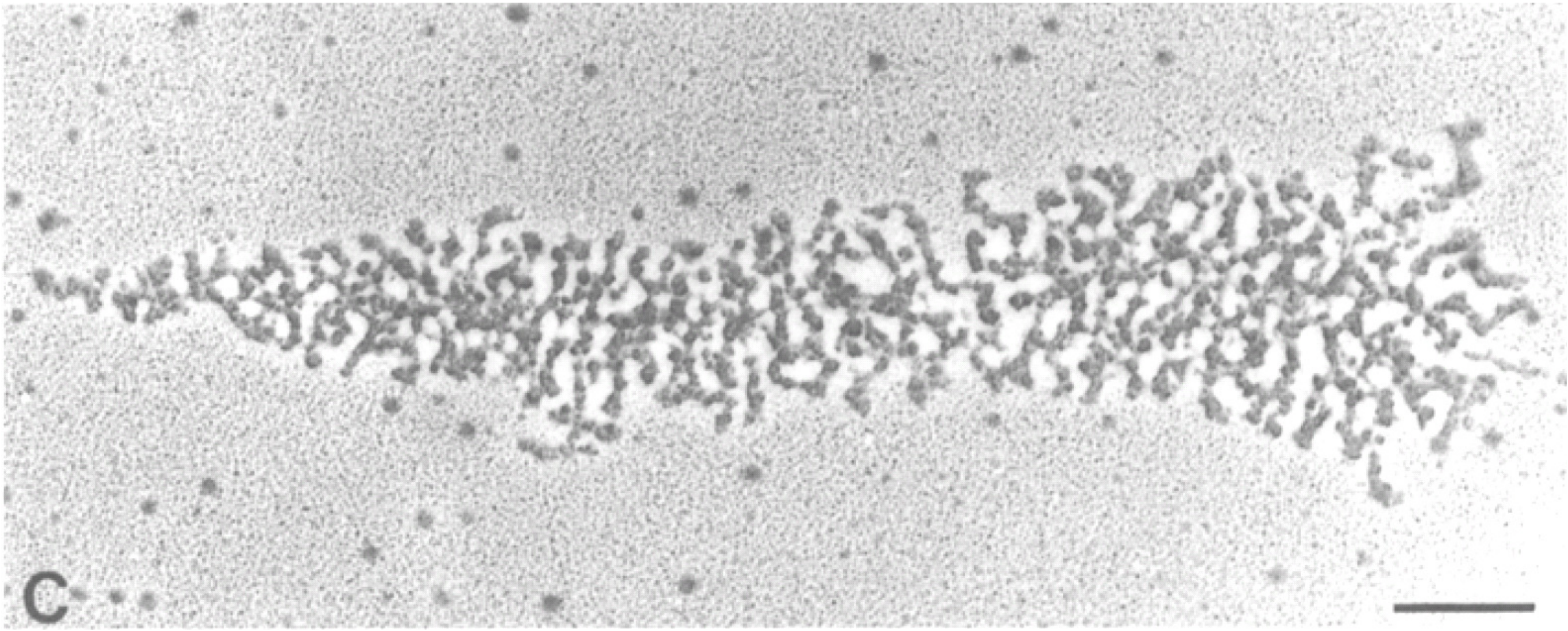
number vertices	40	50	60	70	80
average # edit	116	183	263	352	459
2.42^k no merging	31min	5 h		days...	
3^k with merging	29 s	8 min	27 h	58 h*	19 d*
edge branching	3 s	18 s	6 h	18 h	14 h
with data reduction	1 s	2 s	32 s	43 s	23 s
ILP with data red.			seconds...		

S. Böcker, S. Briesemeister, G. W. Klau. Exact Algorithms for Cluster Editing: Evaluation and Experiments. *Algorithmica* 2009.

Solve instances with thousands of vertices and k > 10000.

Parameterized algorithm vs. ILP





Looking at Christmas trees in the nucleolus
Ulrich Scheer, Bangying Xia, Hilde Merkert, Dieter Weisenberger
Chromosoma (1997) 105:470-480