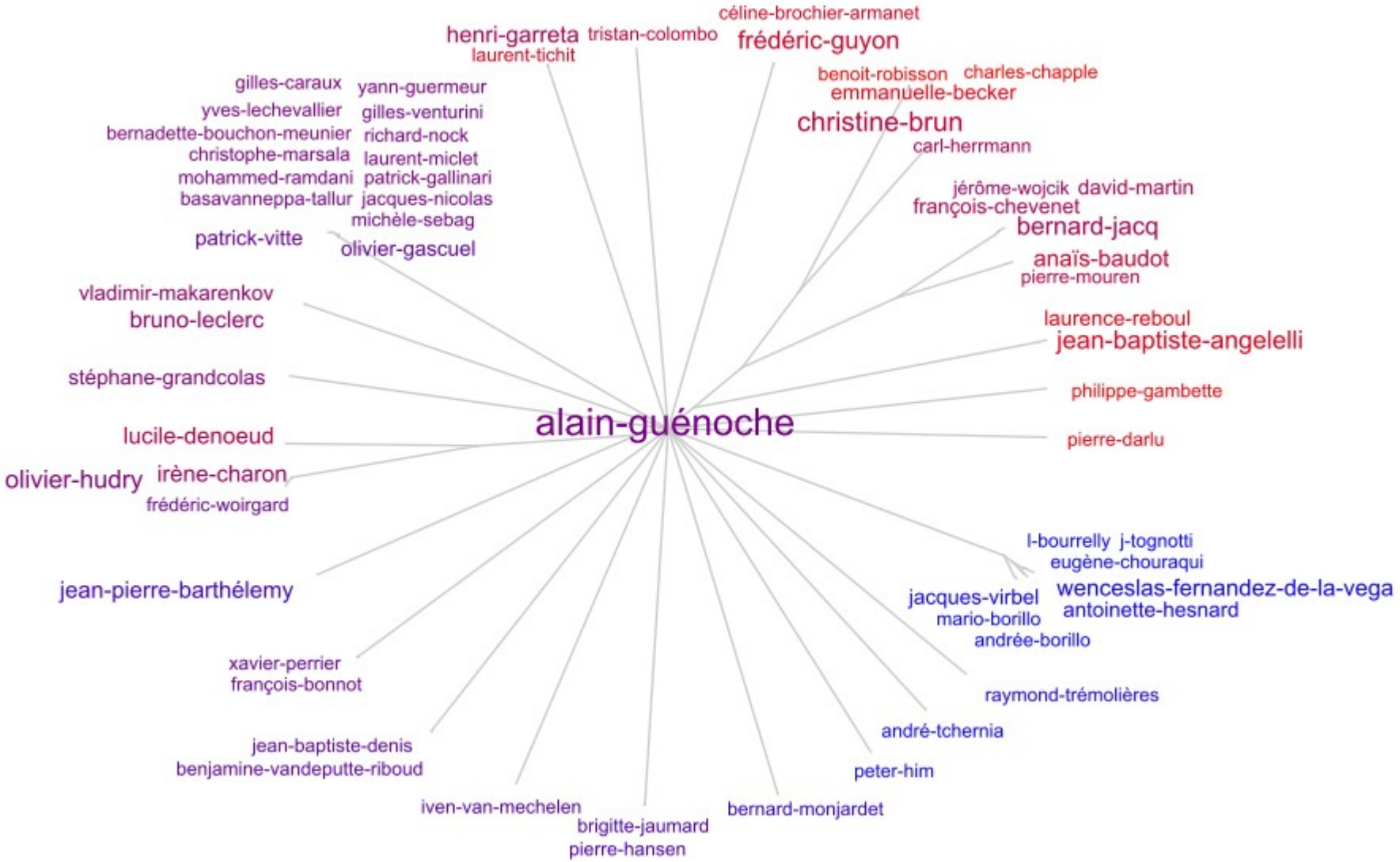
Happy Guénoche Symposium
Marseille – 25/10/2012

The overlap distance between partitions

Philippe Gambette

Joint work with Eunjung Kim (LAMSADE, CNRS)
and Stéphane Thomassé (LIP, ENS Lyon)

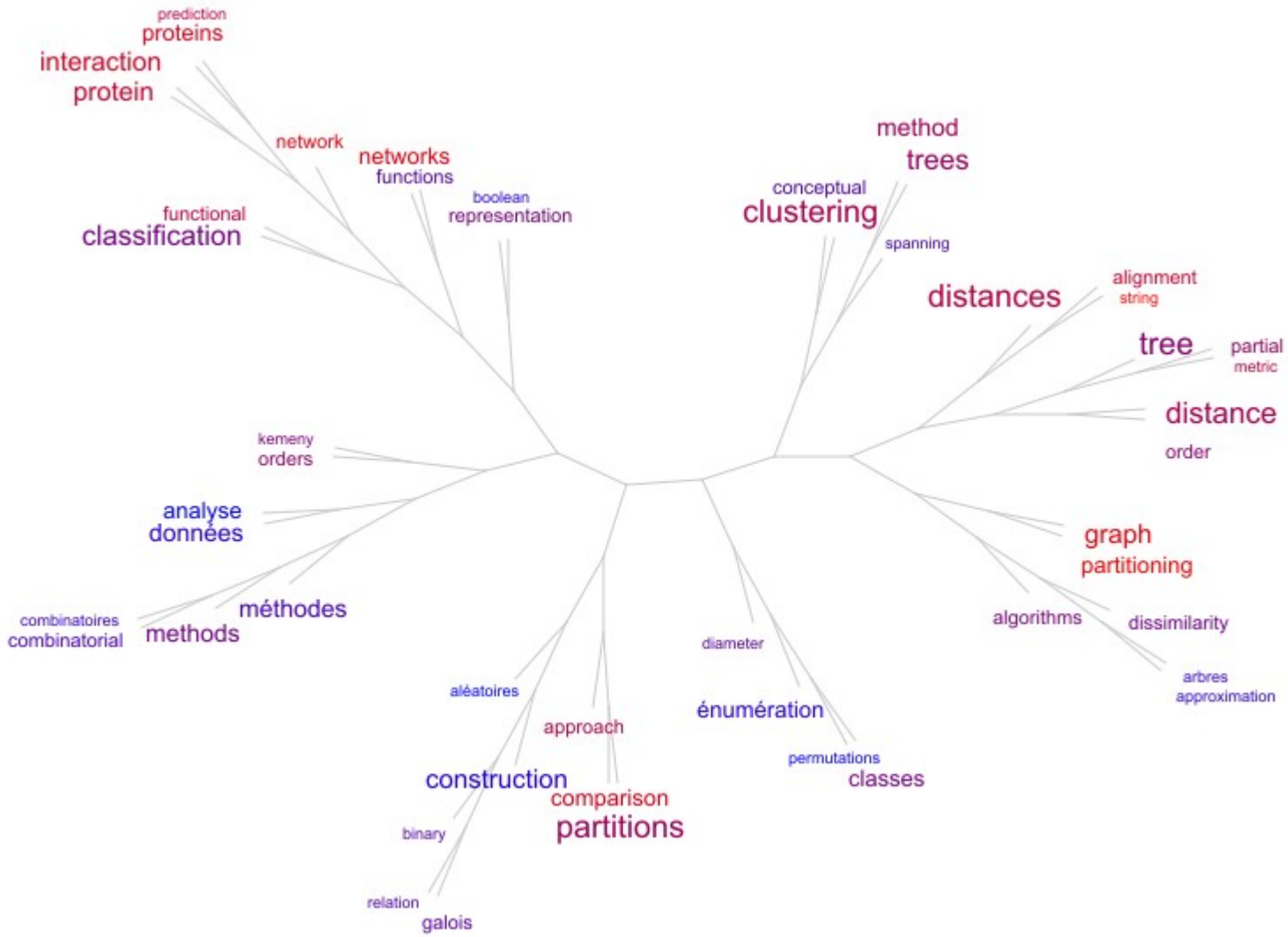
Alain's coauthors



Treecloud of Alain's coauthors (hyperlex cooccurrence distance)

red color: most recent publications

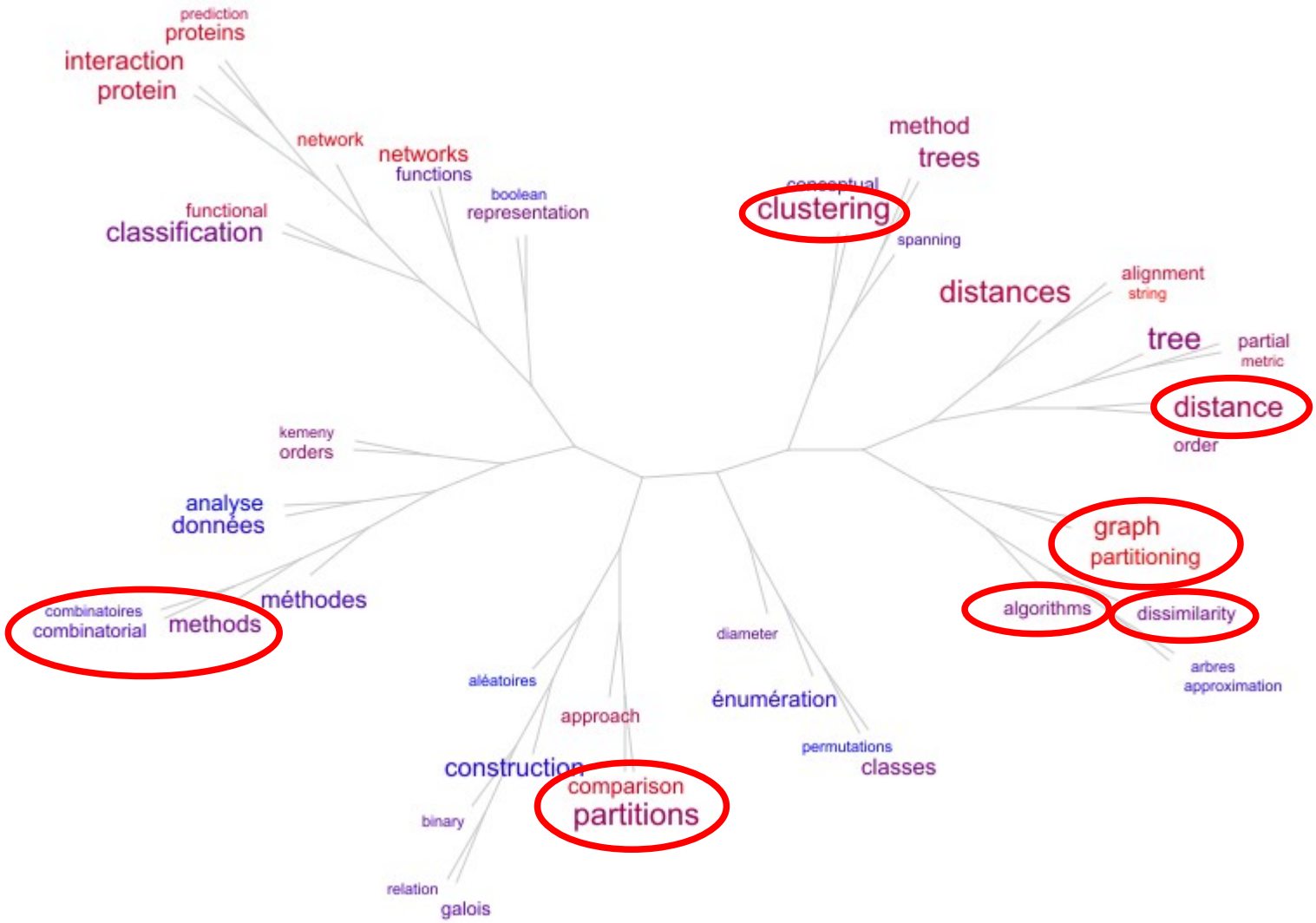
Alain's publications



Treecloud of the 50 most frequent words in Alain's publication titles.

red color: most recent publications

Alain's publications



Treecloud of the 50 most frequent words in Alain's publication titles.

red color: most recent publications

Outline

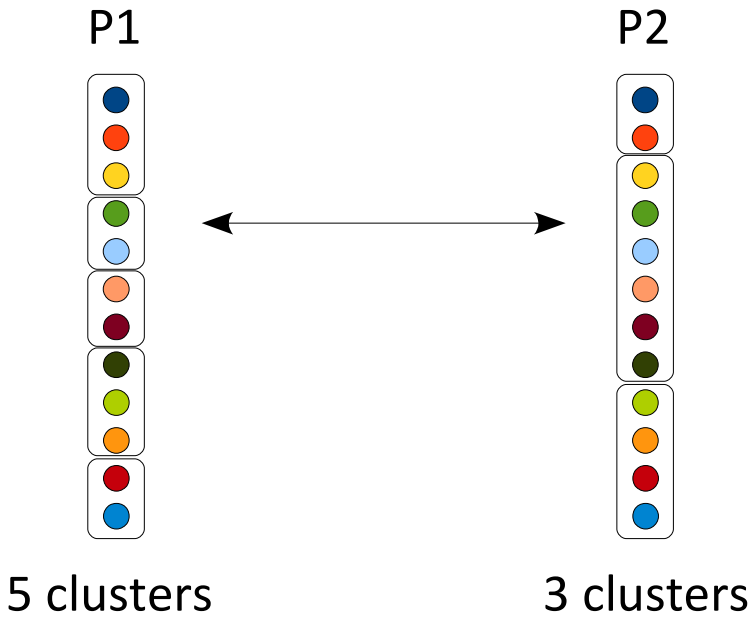
- The overlap distance between partitions
- Equivalent and related problems
- A fixed-parameter complexity approach
- A linear kernel for a restricted version
- Perspectives

Outline

- The overlap distance between partitions
- Equivalent and related problems
- A fixed-parameter complexity approach
- A linear kernel for a restricted version
- Perspectives

Comparing partitions

Problem: **comparing two partitions** P1 and P2

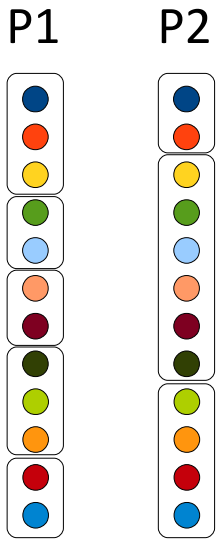


How similar are P1 and P2?

Comparing partitions

Several existing **distances between partitions**:

- Rand distance
- Adjusted Rand distance
- Transfer distance, and more generally:
  Minimum Length Sequence metrics:
  removal, augmentation, mutation, division, mergence, transfer
- ...

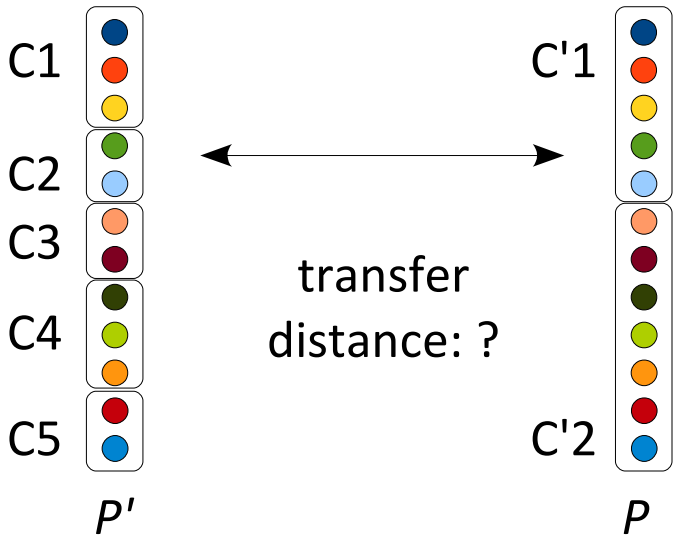


Day, 1981
Guénoche & Denoeud, 2006

Comparing partitions

Several existing **distances between partitions**.

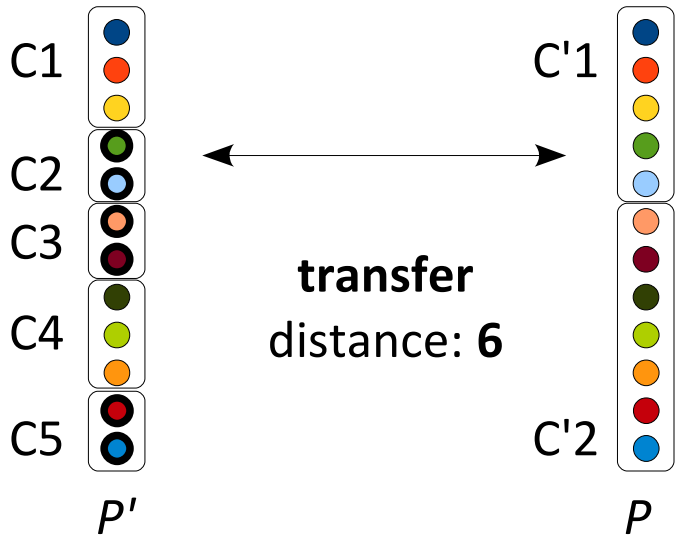
If the number of classes **varies a lot** between two partitions:



Comparing partitions

Several existing **distances between partitions**.

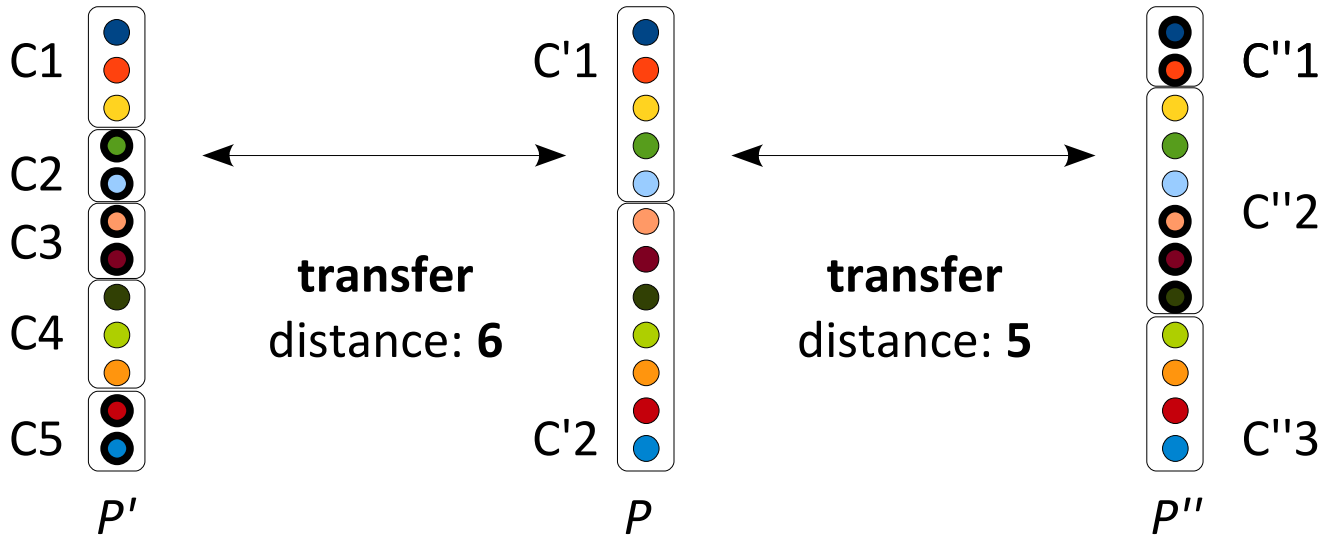
If the number of classes **varies a lot** between two partitions:



Comparing partitions

Several existing **distances between partitions**.

If the number of classes **varies a lot** between two partitions:

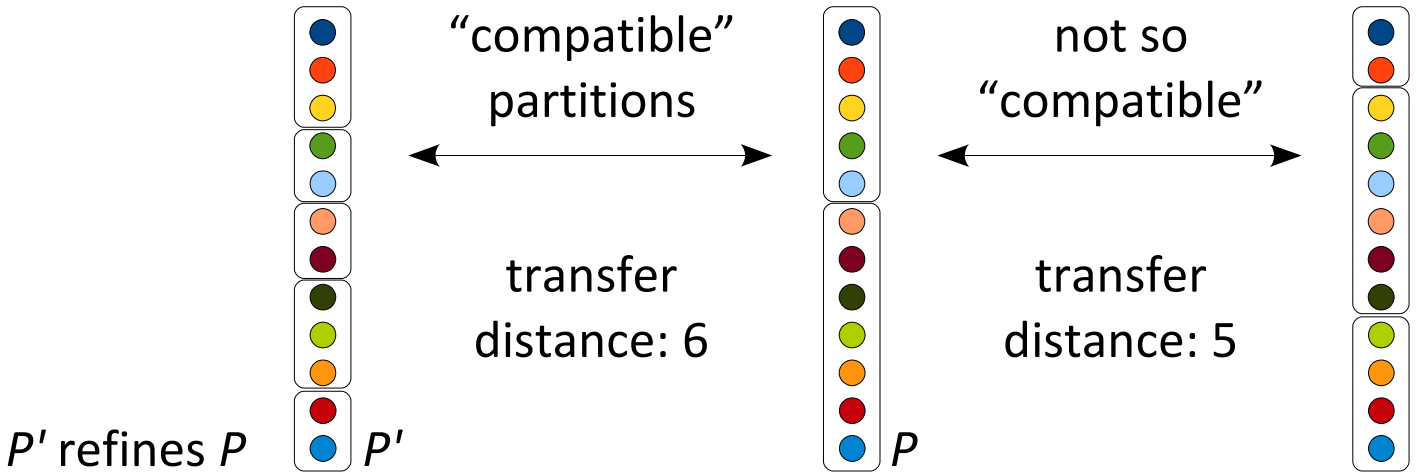


Comparing clustering algorithm results

Several existing **distances between partitions**.

occurs when **comparing two clustering algorithms**, or **two sets of parameters** for the same algorithm

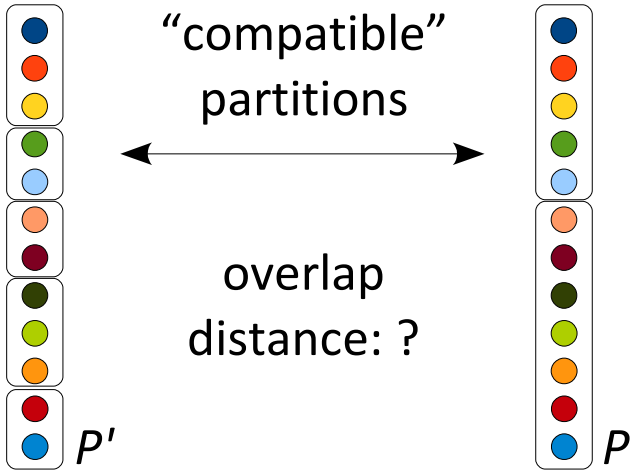
If the number of classes **varies a lot** between two partitions:



A new distance measure between partitions

Overlap distance:

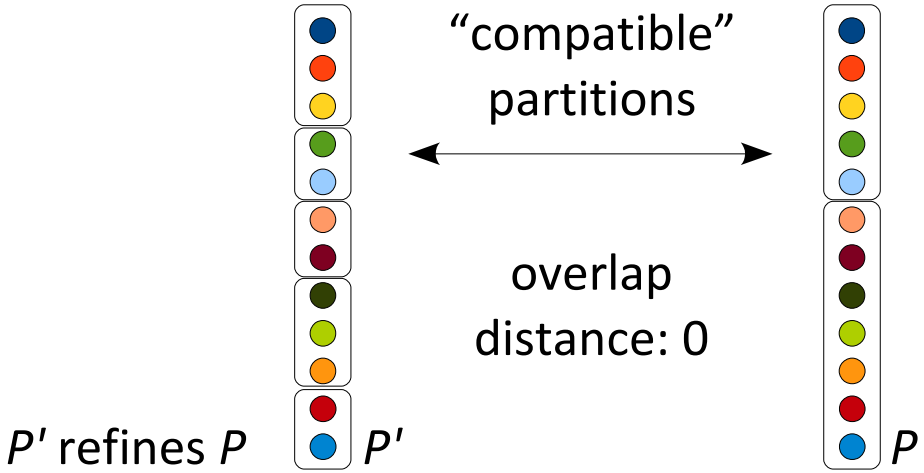
d(P,P') = min |{elements whose deletion remove all overlaps between P and P'}|



A new distance measure between partitions

Overlap distance:

d(P,P') = min |{elements whose deletion remove all overlaps between P and P'}|



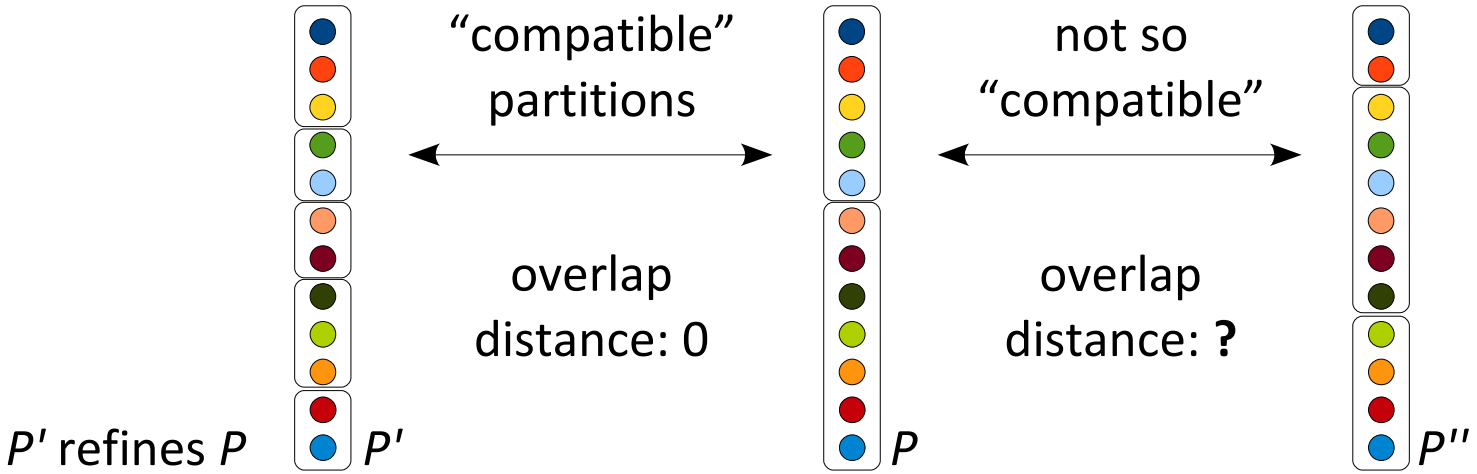
Not a **metric** (no separation property)
just a **dissimilarity**

if no overlap → **easy to visualize both partitions**

A new distance measure between partitions

Overlap distance:

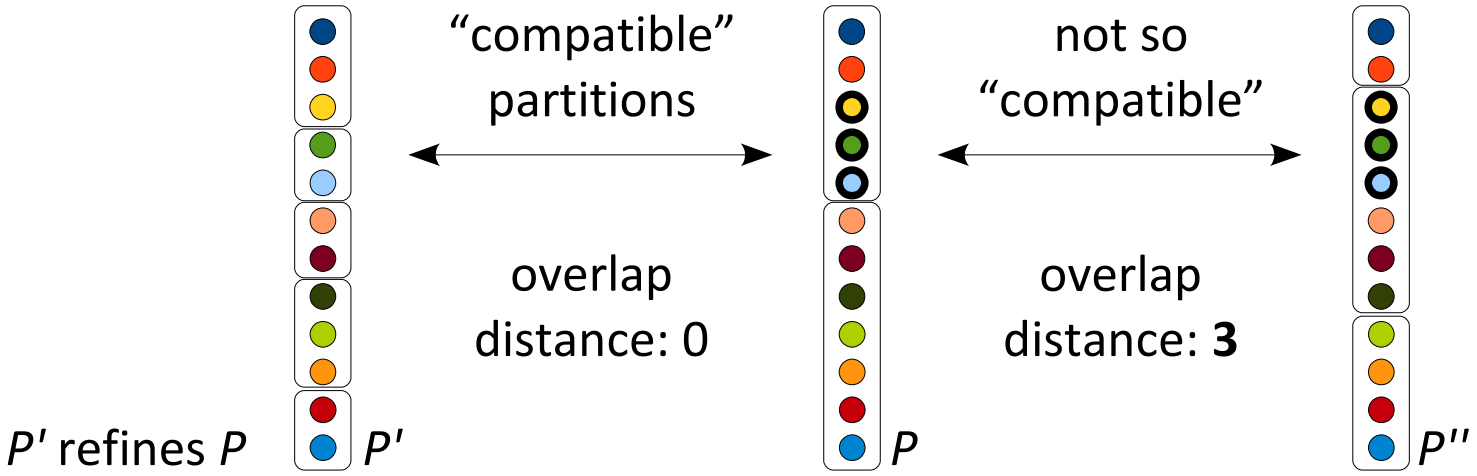
d(*P*,*P'*) = min |{elements whose deletion remove all overlaps between *P* and *P'*}|



A new distance measure between partitions

Overlap distance:

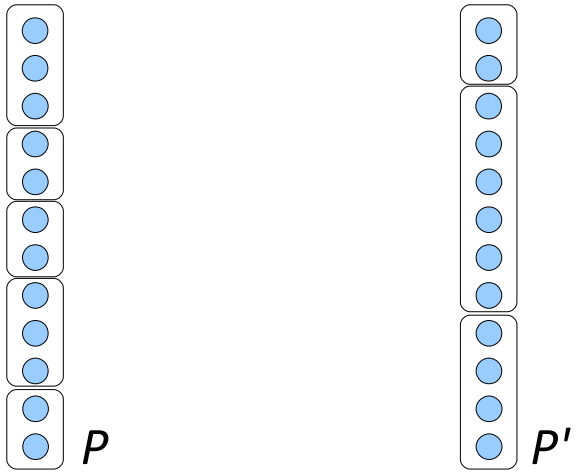
d(*P*,*P'*) = min |{**elements** whose deletion remove all overlaps between *P* and *P'*}|



Another example

Overlap distance:

d(P,P') = min |{elements whose deletion remove all overlaps between P and P'}|

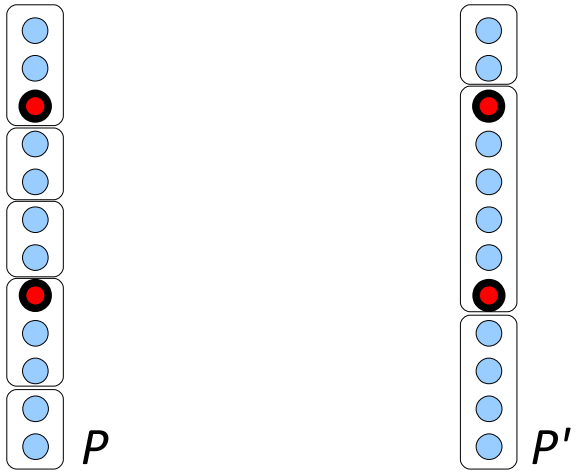


d(P,P') = ?

Another example

Overlap distance:

d(P,P') = min |{**elements** whose deletion remove all overlaps between P and P'}|



d(P,P') = 2

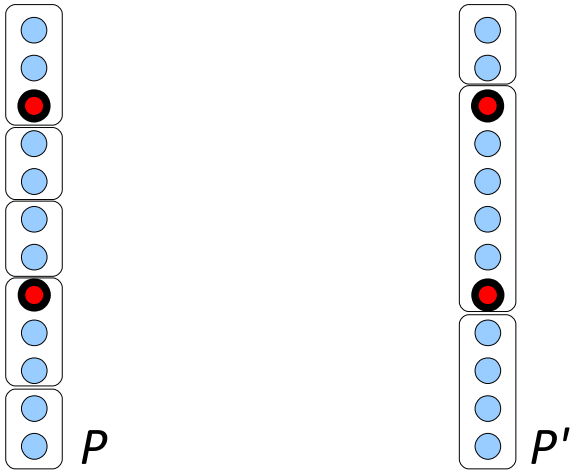
Outline

- The overlap distance between partitions
- Equivalent and related problems
- A fixed-parameter complexity approach
- A linear kernel for a restricted version
- Perspectives

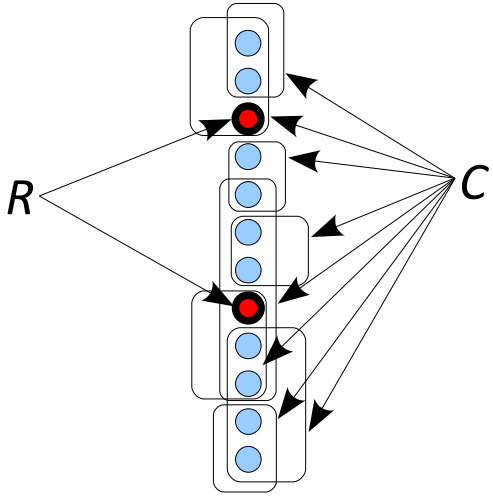
An equivalent problem

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Overlap distance
between *P* and *P'*



Maximum Compatible Subset
of *C* = *P* U *P'*

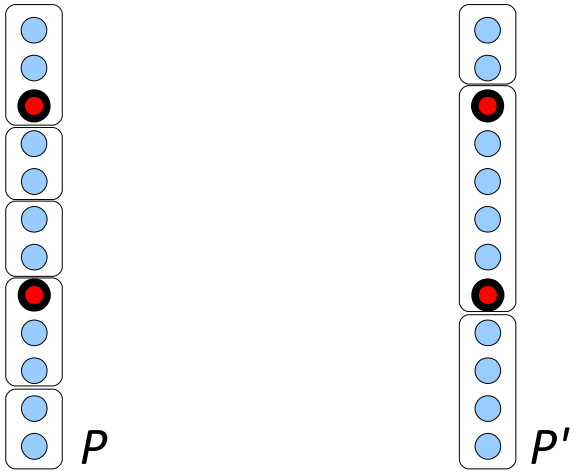


An equivalent problem: restricted Max Compatible Subset

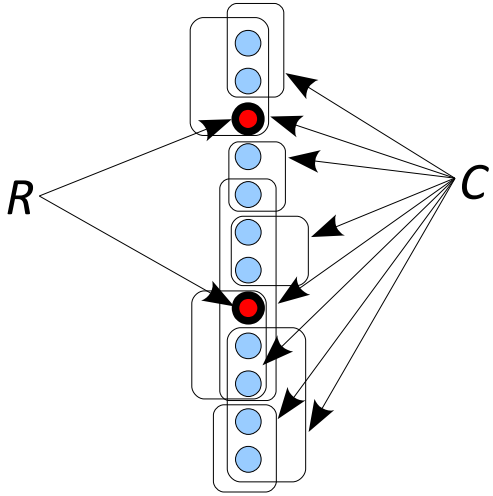
Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Overlap distance = restricted **Maximum Compatible Subset**:
the input clusters come from 2 partitions of *X*

Overlap distance
between *P* and *P'*



Maximum Compatible Subset
of *C* = *P* U *P'*

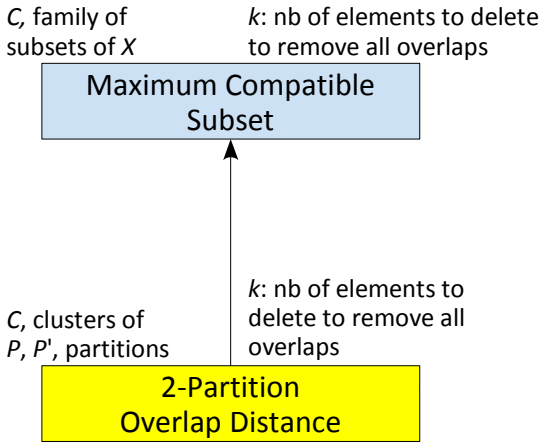


An equivalent problem: restricted Max Compatible Subset

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Overlap distance = restricted **Maximum Compatible Subset**:
the input clusters come from 2 partitions of *X*

Maximum Compatible Subset is NP-hard Steel & Hamel, 1996



An equivalent problem: restricted Max Compatible Subset

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Overlap distance = restricted **Maximum Compatible Subset**:
the input clusters come from 2 partitions of *X*

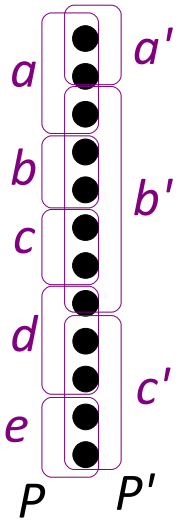
Maximum Compatible Subset is NP-hard Steel & Hamel, 1996

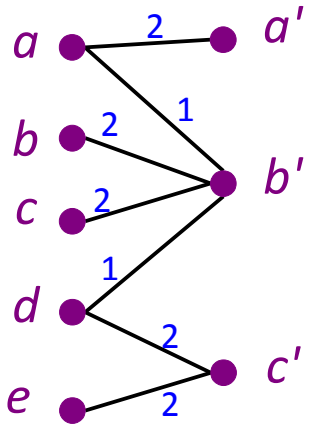
Restricted Maximum Compatible Subset is NP-hard
Gambette, Kim & Thomassé, 2012

Restricted Maximum Compatible Subset is NP-hard

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Cluster intersection graph:





Each vertex: a set of *C* (subset of *P* or *P'*)

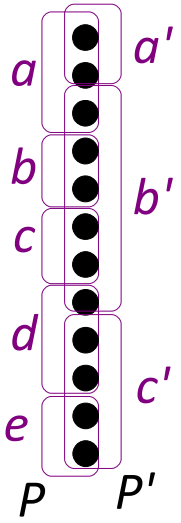
Edge between vertex *x* and *y*
if cluster *x* intersects with cluster *y*

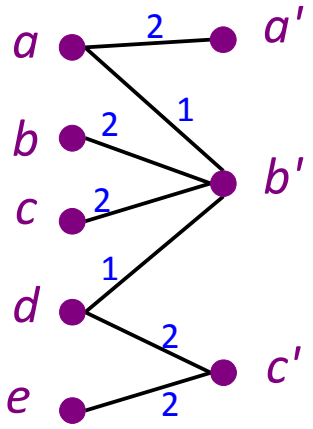
Edge weight = number of elements in the intersection

Restricted Maximum Compatible Subset is NP-hard

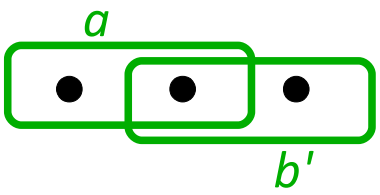
Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Cluster intersection graph:





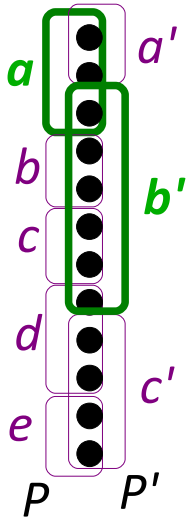
What characterizes two **overlapping clusters** in the cluster intersection graph?

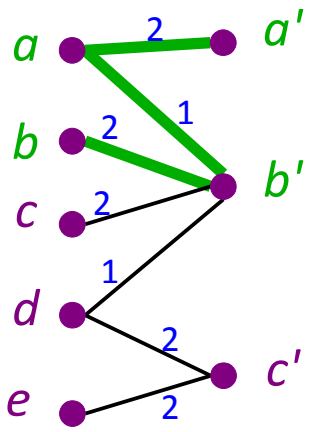


Restricted Maximum Compatible Subset is NP-hard

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Cluster intersection graph:



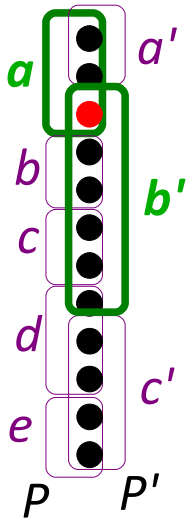


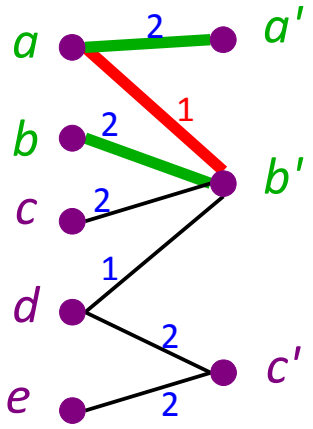
Two **overlapping clusters** (*a* and *b'*) iff a $\mathbf{P_4}$ in the cluster intersection graph

Restricted Maximum Compatible Subset is NP-hard

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Cluster intersection graph:





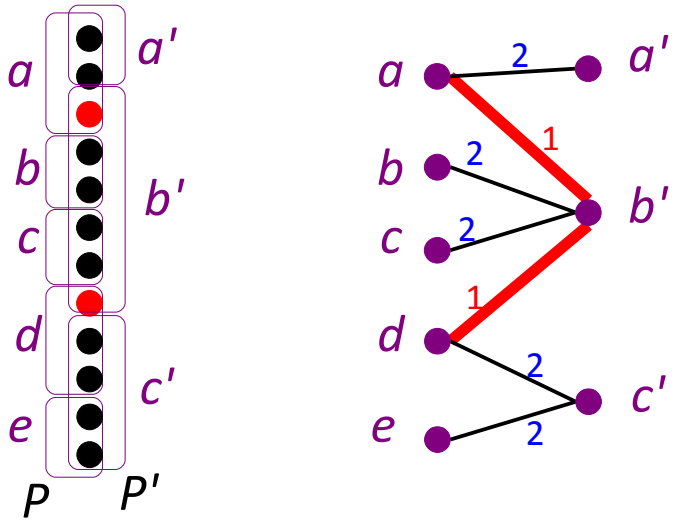
Two **overlapping clusters** (*a* and *b'*) iff a $\mathbf{P_4}$ in the cluster intersection graph

Deleting **elements of *X*** to remove the **overlap** equivalent to destroying the $\mathbf{P_4}$ by removing **an edge**

Restricted Maximum Compatible Subset is NP-hard

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Cluster intersection graph:



Deleting **the minimum number of elements of *X*** to **remove all overlaps**

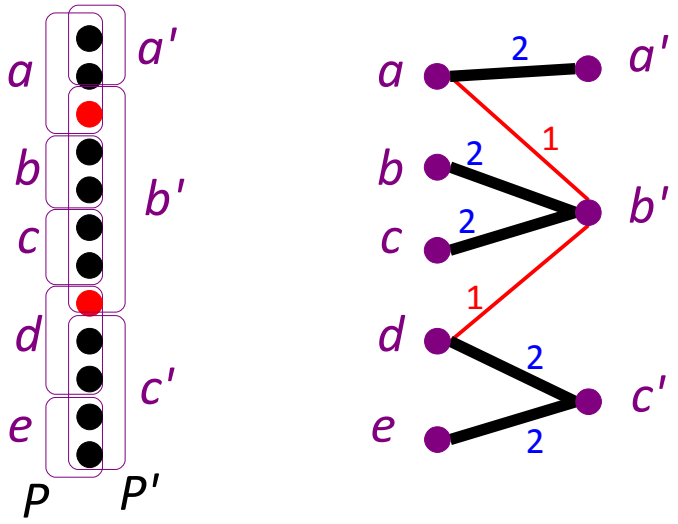
equivalent to

destroying all P_4 graphs by removing a **set of edges of minimum size**

Restricted Maximum Compatible Subset is NP-hard

Maximum Compatible Subset problem :
Input: set *C* of subsets of elements of a set *X*
Output: the smallest subset *R* of *X* such that *C* restricted to *X*-*R* contains no overlap.

Cluster intersection graph:



Deleting **the minimum number of elements of *X*** to **remove all overlaps**

equivalent to

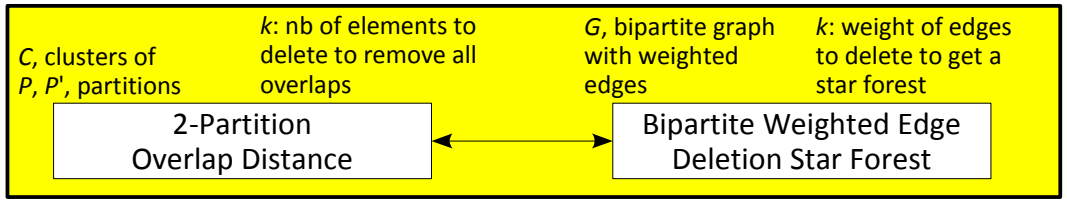
Getting a **star forest** by deleting a **set of edges of minimum weight**
(in a bipartite graph)

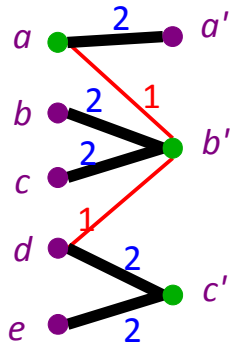
Bipartite Weighted Edge Deletion Star Forest is NP-hard

Bipartite Weighted Edge Deletion Star Forest problem :
Input: bipartite graph *G*
Output: a subset *E'* of edges of *G* of minimum weight such that *G*-*E'* is a star forest

Restricted Maximum Compatible Subset is equivalent to **Bipartite Weighted Edge Deletion Star Forest**.

Gambette, Kim & Thomassé, 2012





Bipartite weighted edge deletion star forest is NP-hard.

Chen, Engelberg et al 2007

The more restricted case where all weights equal 1 is equivalent to finding a **minimum dominating set** in a bipartite graph → NP-hard

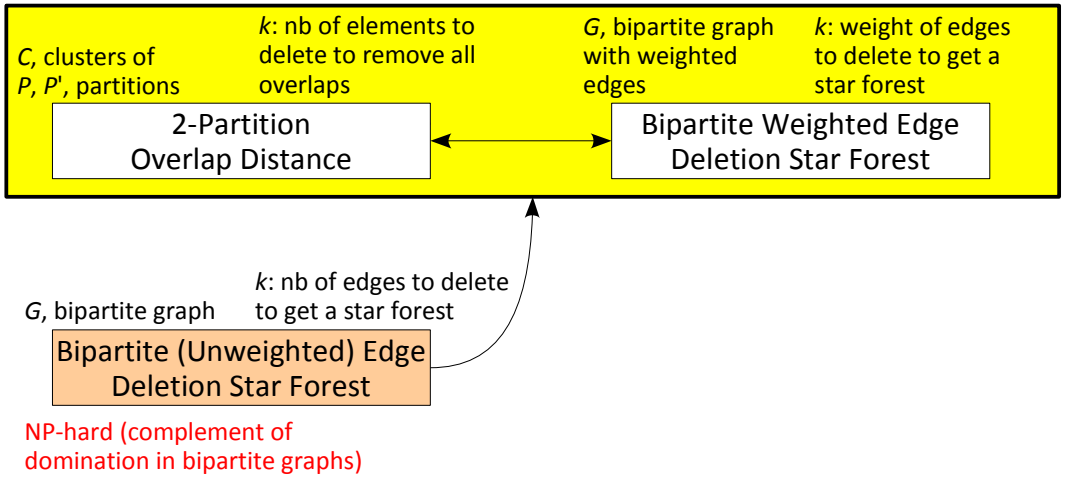
ISGCI (http://www.graphclasses.org/classes/gc_69.html), Dewdney 1981

Bipartite Weighted Edge Deletion Star Forest is NP-hard

Bipartite Weighted Edge Deletion Star Forest problem :
Input: bipartite graph G
Output: a subset E' of edges of G of minimum weight such that $G-E'$ is a star forest

Restricted Maximum Compatible Subset is equivalent to **Bipartite Weighted Edge Deletion Star Forest**.

Gambette, Kim & Thomassé, 2012



Outline

- The overlap distance between partitions
- Equivalent and related problems
- A fixed-parameter complexity approach
- A linear kernel for a restricted version
- Perspectives

Fixed-parameter complexity

Computing the overlap distance: NP-hard

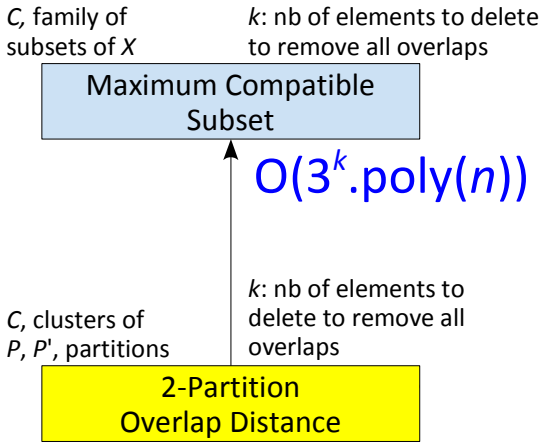
Fixed-parameter complexity approach:
Deciding whether the **overlap distance is at most *k*** in time **O(*f*(*k*)*poly(*n*))**
(function *f* can grow exponentially fast)

$O^*(3^k)$ by reducing to Maximum Compatible Subset

Computing the overlap distance: NP-hard

Fixed-parameter complexity approach:
Deciding whether the **overlap distance is at most *k*** in time **$O(f(k)*\text{poly}(n))$**
(function *f* can grow exponentially fast)

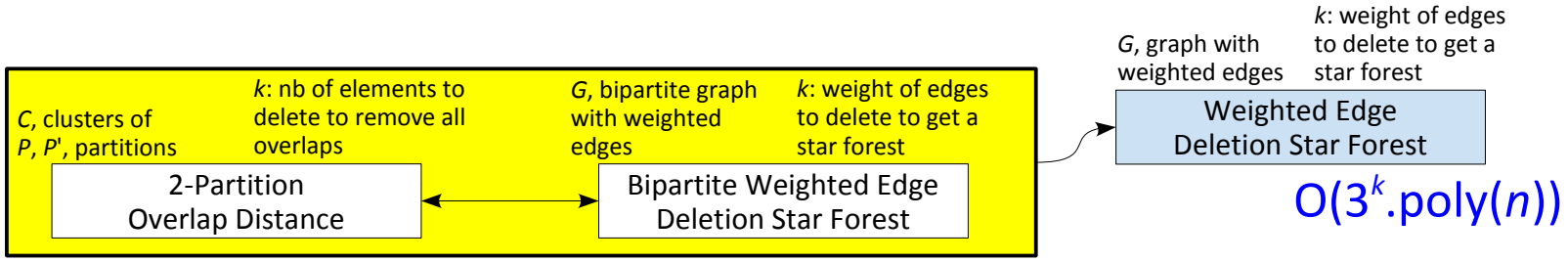
Maximum Compatible Subset can be solved in time $O(3^k.\text{poly}(n))$
Huson, Rupp, Berry, Gambette & Paul, 2009



$O^*(3^k)$ by reducing to Weighted Edge Deletion Star Forest

Computing the overlap distance: NP-hard

Fixed-parameter complexity approach:
Deciding whether the **overlap distance is at most *k*** in time **O(*f*(*k*)*poly(*n*))**
(function *f* can grow exponentially fast)



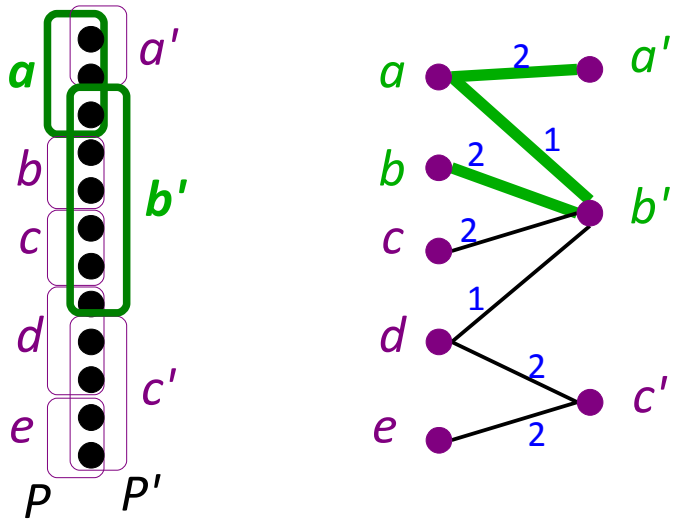
$O^*(3^k)$ by reducing to Weighted Edge Deletion Star Forest

Computing the overlap distance: NP-hard

Fixed-parameter complexity approach:
Deciding whether the **overlap distance is at most *k*** in time **$O(f(k)*poly(n))$**
(function *f* can grow exponentially fast)

Easy $O(3^k.poly(n))$ time algorithm

Cluster intersection graph:



Two **overlapping clusters** (*a* and *b'*)
iff a $\mathbf{P_4}$ in the cluster intersection graph

Try all possibilities to remove the $\mathbf{P_4}$.

You **have to** remove either
- first edge *a*-*a'* (cost 2)
- second edge *a*-*b'* (cost 1)
- third edge *b*-*b'* (cost 2)

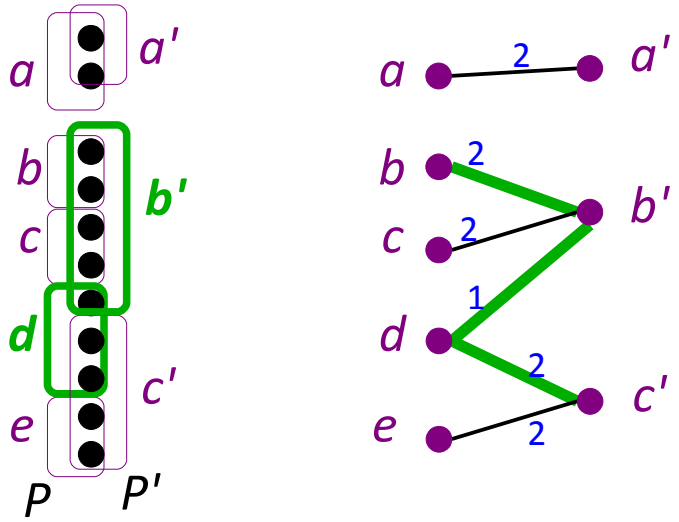
$O^*(3^k)$ by reducing to Weighted Edge Deletion Star Forest

Computing the overlap distance: NP-hard

Fixed-parameter complexity approach:
Deciding whether the **overlap distance is at most *k*** in time **$O(f(k)*\text{poly}(n))$**
(function *f* can grow exponentially fast)

Easy $O(3^k.\text{poly}(n))$ time algorithm

Cluster intersection graph:



Edge *a*-*b'* (cost 1) removed

Consider **the next P_4** and **repeat until the total cost reaches *k***.

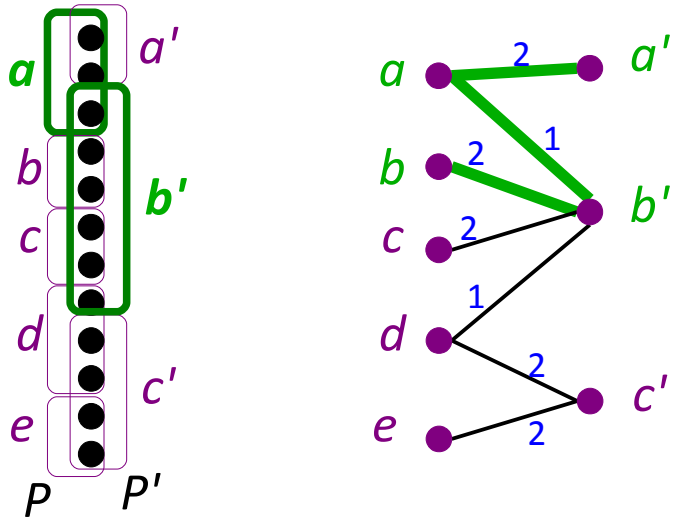
→ "bounded search tree" of height *k*
→ $O(3^k.\text{poly}(n))$ time

$O^*(2.247^k)$ by reducing to Weighted 3-Hitting Set

$O(2.247^k.\text{poly}(n))$ time algorithm using Weighted 3-Hitting Set

Weighted 3-Hitting Set problem :
Input: a set S of weighted elements + a set Y of subsets of S of size 3
Output: a subset V of S of minimum weight such that each subset of S in Y contains at least one element of V

Cluster intersection graph:



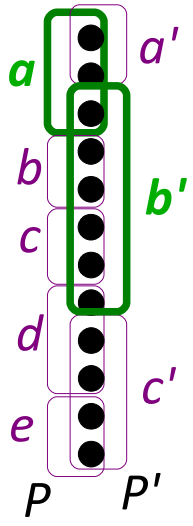
Each P_4 corresponds to a **subset of 3 edges**

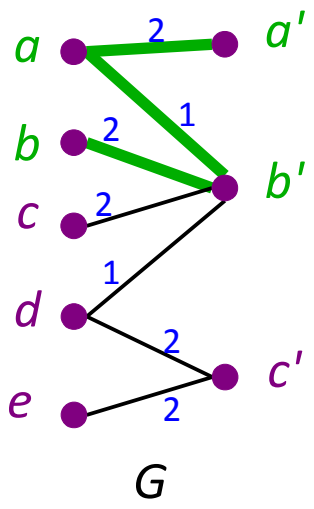
$O^*(2.247^k)$ by reducing to Weighted 3-Hitting Set

$O(2.247^k.\text{poly}(n))$ time algorithm using Weighted 3-Hitting Set

Weighted 3-Hitting Set problem :
Input: a set *S* of weighted elements + a set *Y* of subsets of *S* of size 3
Output: a subset *V* of *S* of minimum weight such that each subset of *S* in *Y* contains at least one element of *V*

Cluster intersection graph:





Each P_4 corresponds to a **subset of 3 edges**

Removing all P_4 by deleting the subset of edges of minimum weight

reduces to

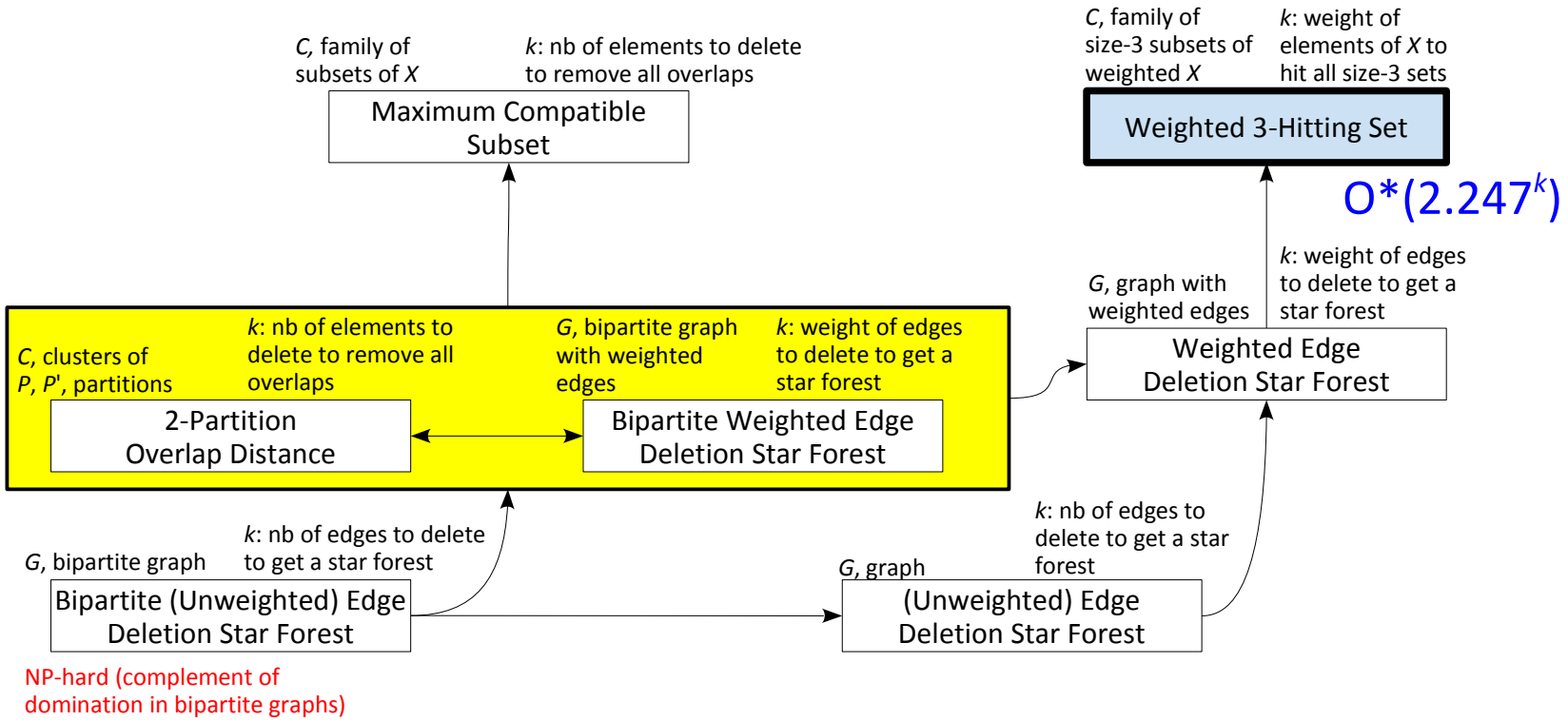
Solving **Weighted 3-Hitting Set**
where *S* is the set of weighted edges of *G*
and *Y* is the set of all P_4 graphs of *G*
(size of *Y*: at most cubic in the size of *C*)

O*(2.247^k) by reducing to Weighted 3-Hitting Set

$O(2.247^k.\text{poly}(n))$ time algorithm using Weighted 3-Hitting Set

Fernau, 2006

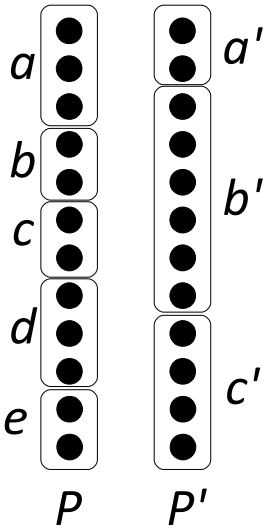
Weighted 3-Hitting Set problem :
Input: a set *S* of weighted elements + a set *Y* of subsets of *S* of size 3
Output: a subset *V* of *S* of minimum weight such that each subset of *S* in *Y* contains at least one element of *V*

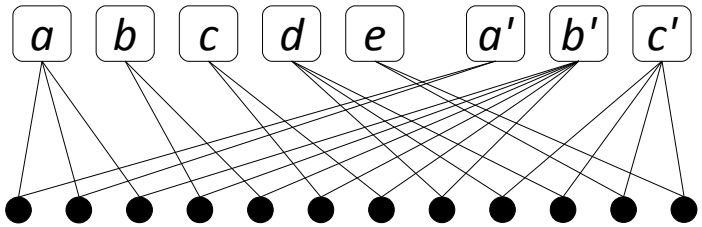


O*(2.076^k) by reducing to 3-Hitting Set

3-Hitting Set problem :
Input: a set *S* of elements + a set *Y* of subsets of *S* of size 3
Output: a subset *V* of *S* of minimum size such that each subset of *S* in *Y* contains at least one element of *V*

Character graph :

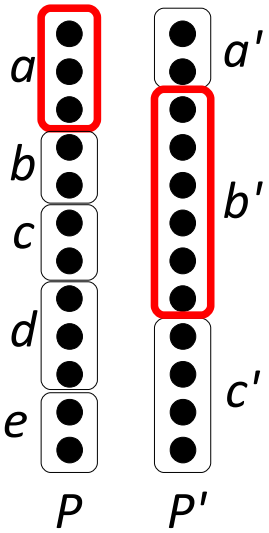


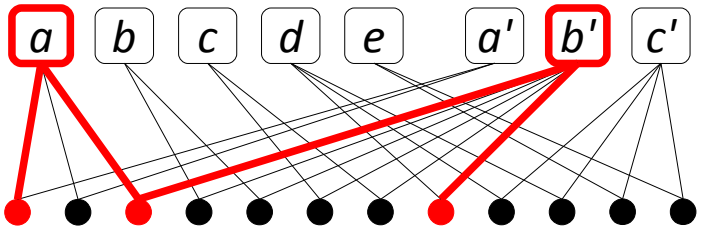


$O^*(2.076^k)$ by reducing to 3-Hitting Set

3-Hitting Set problem :
Input: a set *S* of elements + a set *Y* of subsets of *S* of size 3
Output: a subset *V* of *S* of minimum size such that each subset of *S* in *Y* contains at least one element of *V*

Character graph :



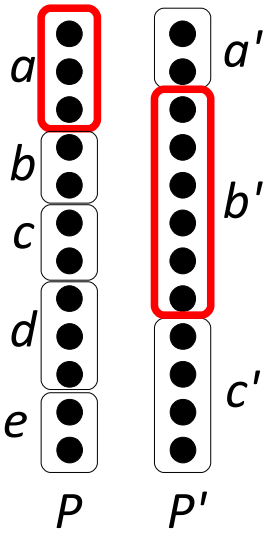


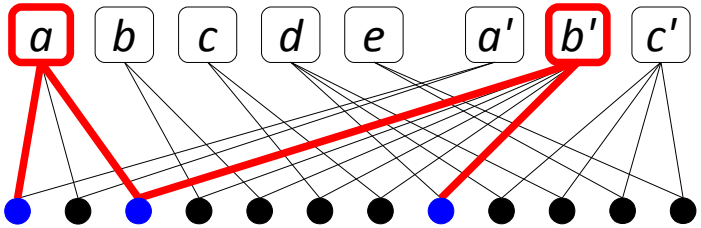
Two clusters **overlap** (*a* and *b'*)
if and only if
the character graph **"contains an M"** as an induced subgraph

$O^*(2.076^k)$ by reducing to 3-Hitting Set

3-Hitting Set problem :
Input: a set *S* of elements + a set *Y* of subsets of *S* of size 3
Output: a subset *V* of *S* of minimum size such that each subset of *S* in *Y* contains at least one element of *V*

Character graph :



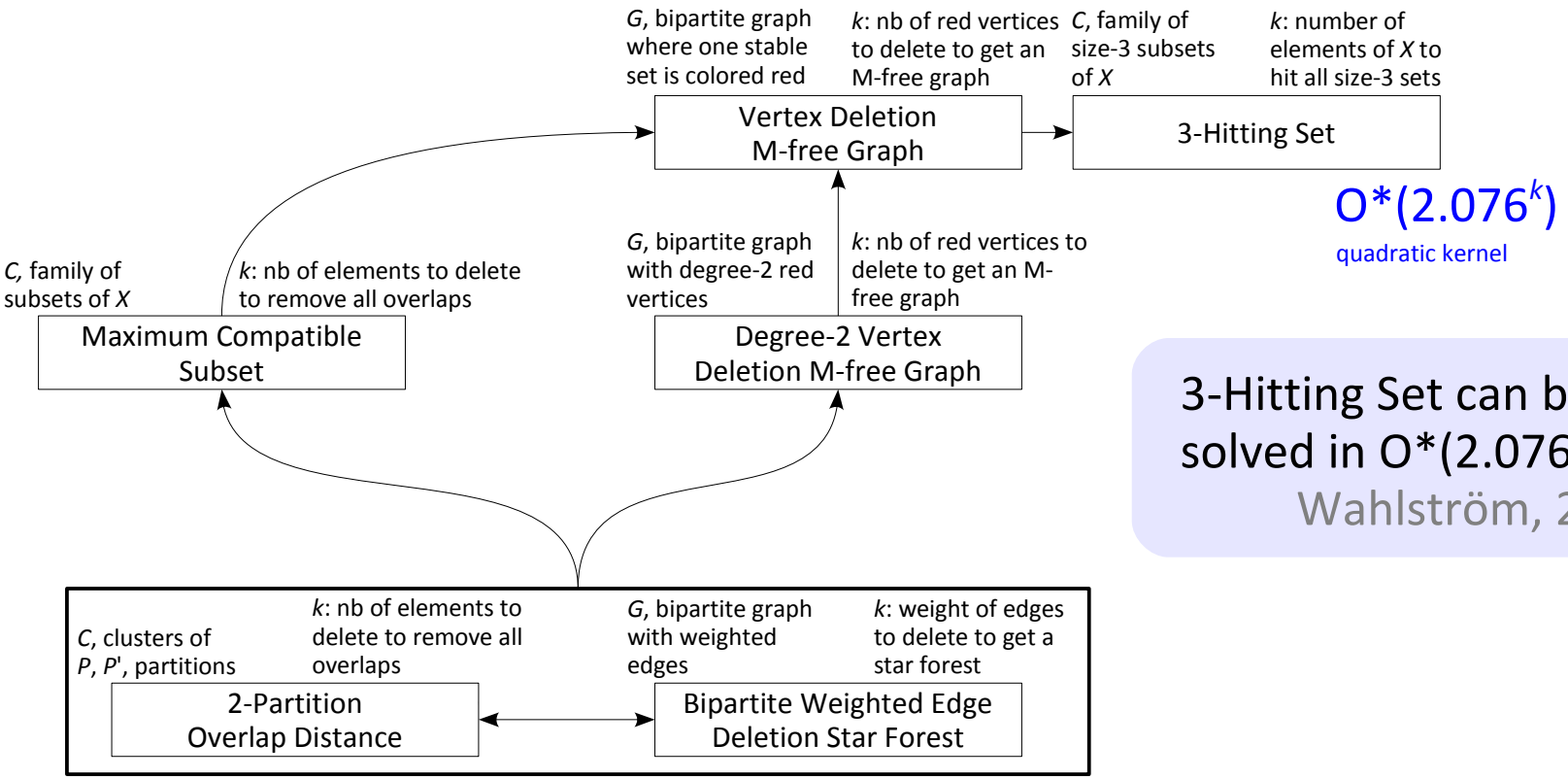


Two clusters **overlap** (*a* and *b'*)
if and only if
the character graph **"contains an M"** as an induced subgraph

Computing the overlap distance *reduces to*
Deleting the minimum nb of vertices to destroy all M-graphs
which reduces to
3-Hitting Set on the **triplets of vertices involved in an M-graph**
Guillemot, 2010

O*(2.076^k) by reducing to 3-Hitting Set

3-Hitting Set problem :
Input: a set *S* of elements + a set *Y* of subsets of *S* of size 3
Output: a subset *V* of *S* of minimum size such that each subset of *S* in *Y* contains at least one element of *V*



3-Hitting Set can be solved in O*(2.076^k)
Wahlström, 2007

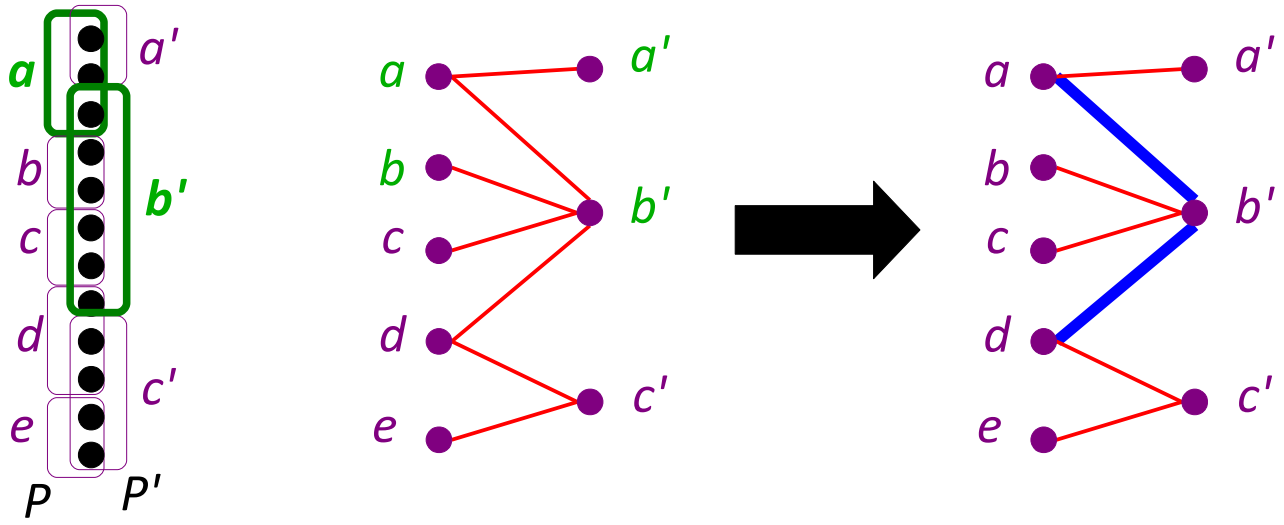
$O^*(1.84^k)$ by reducing to Star Editing (unweighted case)

$O(1.84^k.\text{poly}(n))$ time algorithm using **Star Editing**

Damaschke & Molokov, 2012

Star Editing problem :
Input: a graph with red and blue edges
Output: a **subset *E'* of *E* of minimum size** whose color has to be changed so that the red edges become a union of stars

Cluster intersection (multi)-graph:

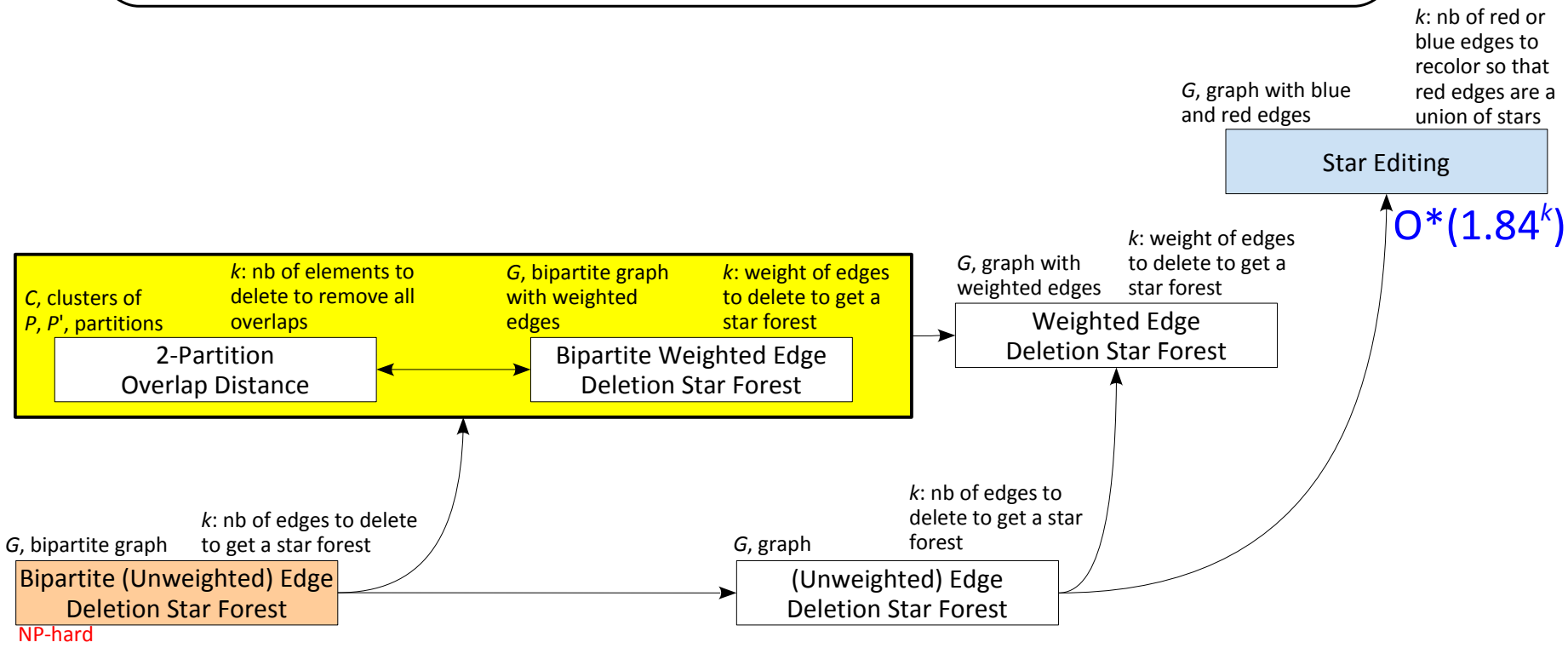


O*(1.84^k) by reducing to Star Editing (unweighted case)

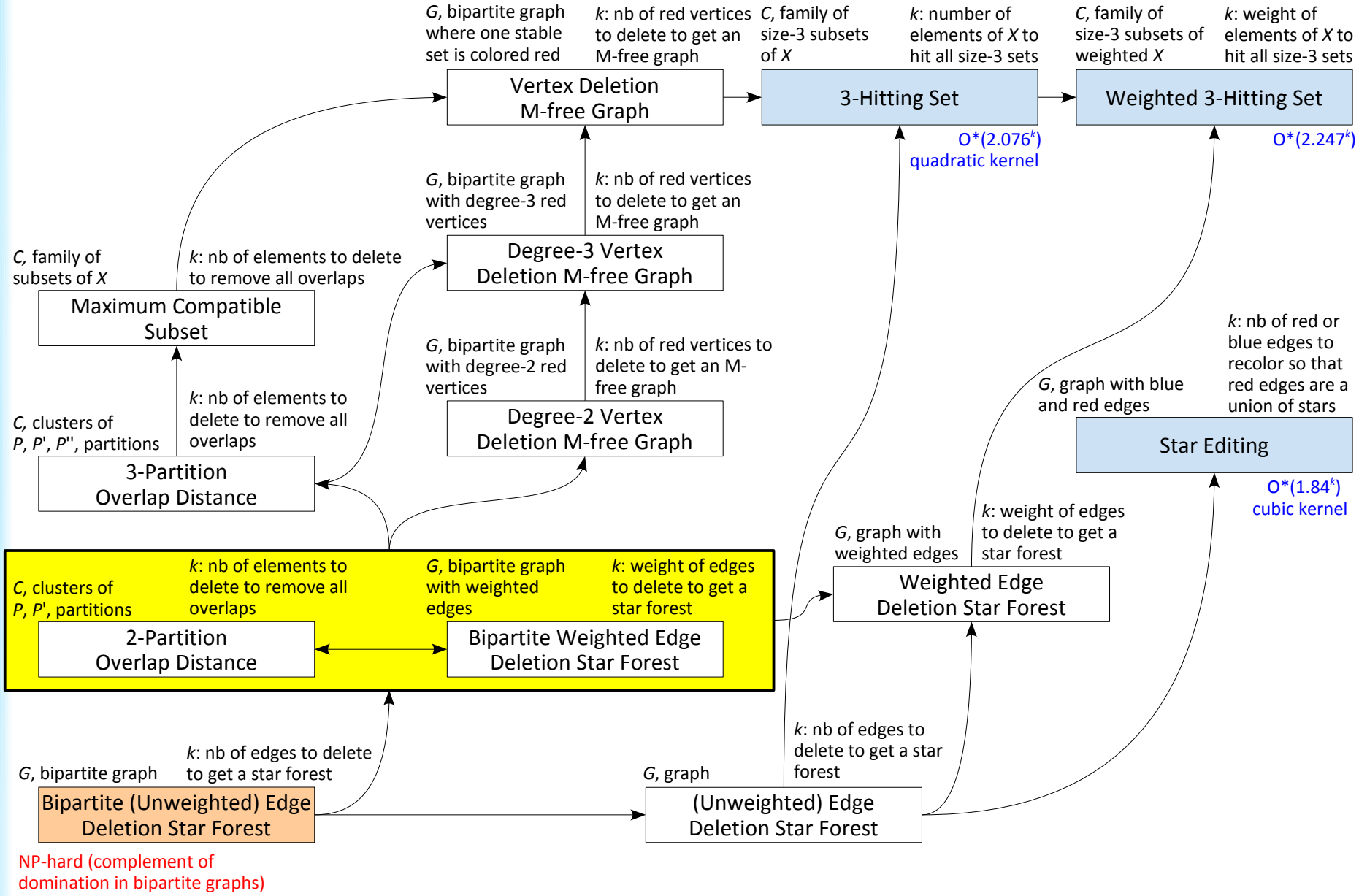
$O(1.84^k.poly(n))$ time algorithm using **Star Editing**

Damaschke & Molokov, 2012

Star Editing problem :
Input: a graph with red and blue edges
Output: a subset E' of E of minimum size whose color has to be changed so that the red edges become a union of stars



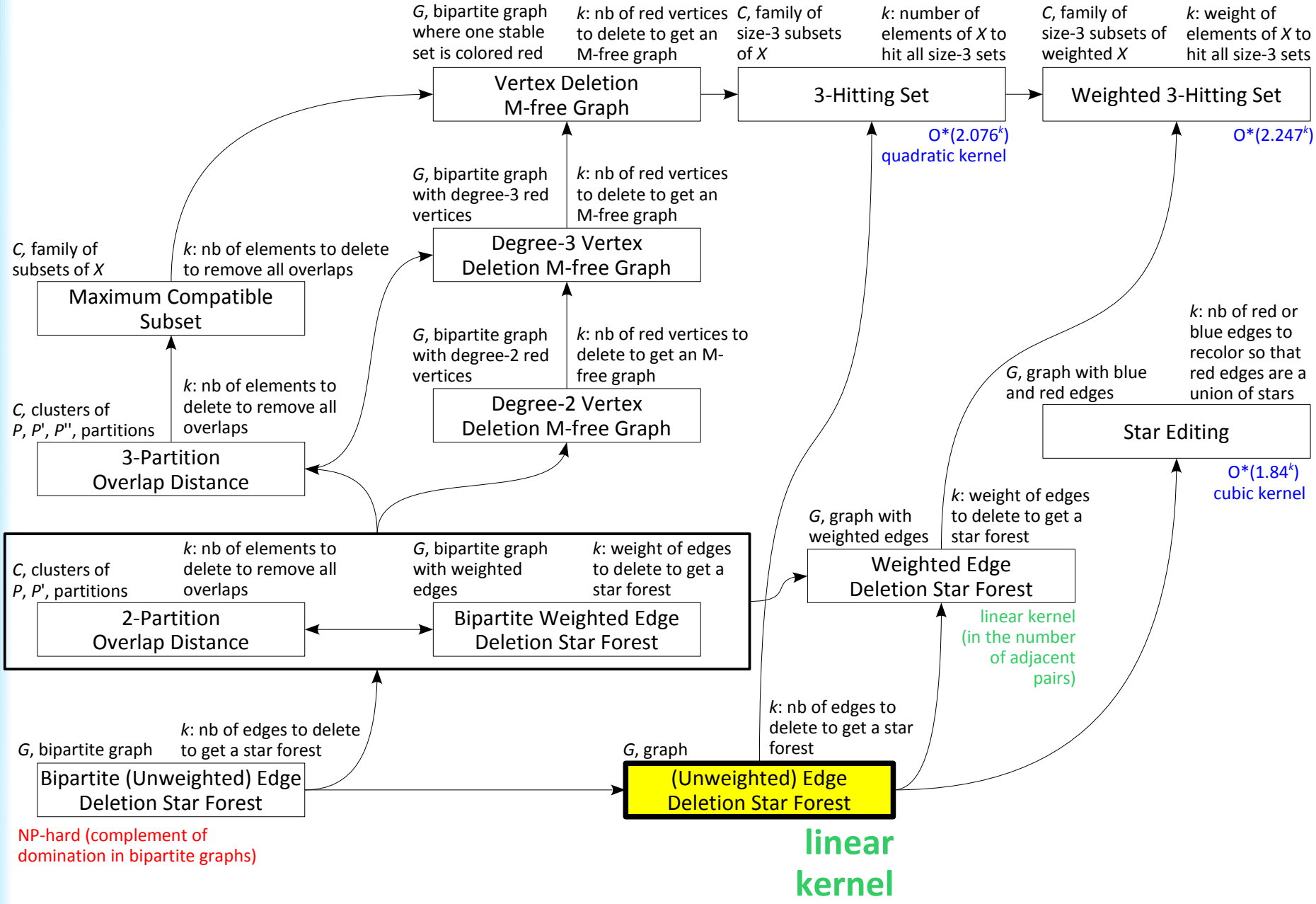
The overlap distance and related problems



Outline

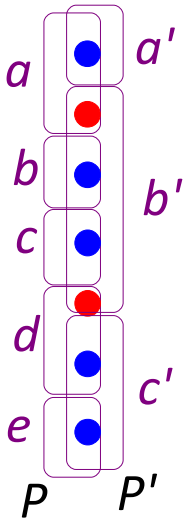
- The overlap distance between partitions
- Equivalent and related problems
- A fixed-parameter complexity approach
- A linear kernel for a restricted version
- Perspectives

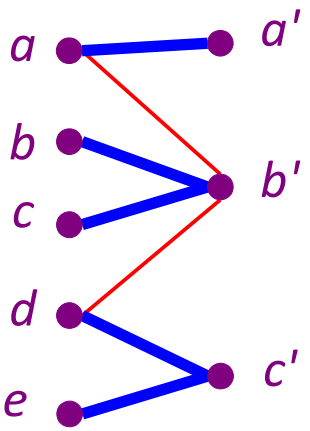
The overlap distance and related problems



Fixed-parameter complexity and kernels

Edge Deletion Star Forest problem :
Input: graph G
Output: a subset E' of edges of G of minimum size k such that G-E' is a **star forest**





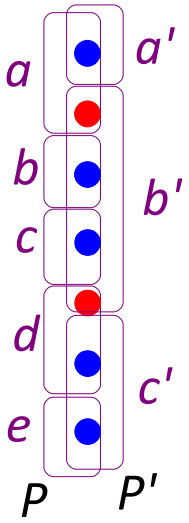
cluster intersection graph G

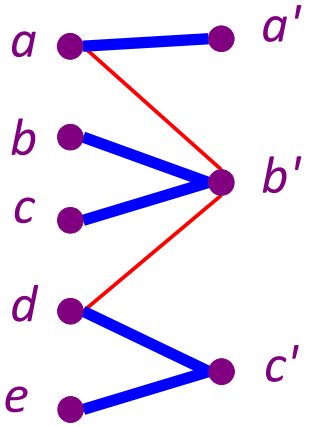
Fixed-parameter complexity and kernels

Edge Deletion Star Forest problem :
Input: graph G
Output: a subset E' of edges of G of minimum size k such that G-E' is a **star forest**

Edge Deletion Star Forest has a kernel of size $30k$

Gambette, Kim & Thomassé, 2012





cluster intersection graph G

"Reduction rules"

→ solve the problem on a smaller graph.

→ after applying the reduction rules, graph with at most $30k$ vertices

→ "Kernel" of size $30k$

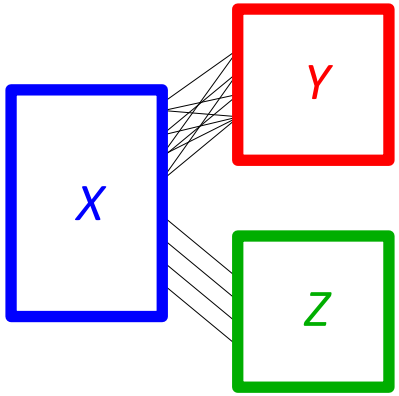
Fixed-parameter complexity and kernels

Edge Deletion Star Forest problem :
Input: graph G
Output: a subset E' of edges of G of minimum size k such that G-E' is a star forest

Edge Deletion Star Forest has a kernel of size $30k$

Gambette, Kim & Thomassé, 2012

(*X*,*Y*,*Z*)-decomposition of G:



- ***X*** is a vertex cover of G of size $\leq 4k$
(can be found easily with Maximum Matching)
- other vertices: in ***Z*** if degree 1, in ***Y*** if >1

"Reduction rules"

→ solve the problem on a smaller graph.

→ after applying the reduction rules, graph with at most $30k$ vertices

→ "Kernel" of size $30k$

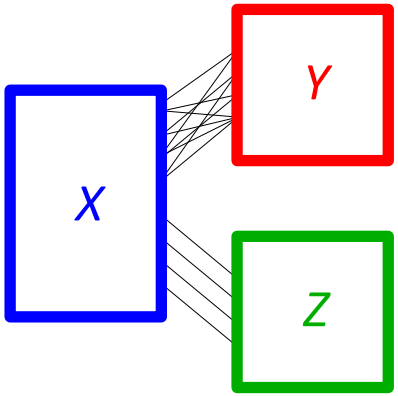
Fixed-parameter complexity and kernels

Edge Deletion Star Forest problem :
Input: graph G
Output: a subset E' of edges of G of minimum size k such that G-E' is a star forest

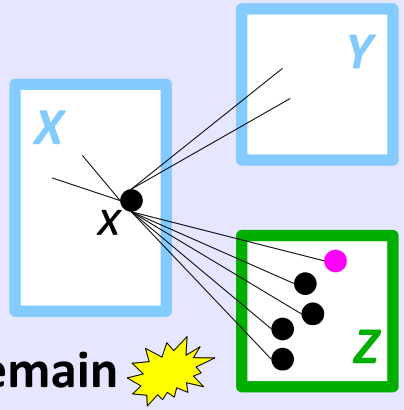
Edge Deletion Star Forest has a kernel of size $30k$

Gambette, Kim & Thomassé, 2012

(*X*,*Y*,*Z*)-decomposition of *G*:



Reduction rule:
As long as a vertex *x* of *X* has more **neighbors in *Z*** than in ***X* U *Y***, remove **one of its neighbors in *Z***



→ **Only $30k$ vertices remain**

Outline

- The overlap distance between partitions
- Equivalent and related problems
- A fixed-parameter complexity approach
- A linear kernel for a restricted version
- Perspectives

Perspectives

• Optimizing (in theory and in practice) the computation of the **overlap distance** (fixed parameter algorithms, approximations, heuristics...):
implementation and comparison on real data

• Analyzing the **properties of this distance measure** and **relevant applications**

• Using the overlap distance to **visualize conflicting partitions**

Thank you!

Especially to Alain, Anaïs, Christine, Elisabeth, Gilles, Laurent for their welcome in Marseilles