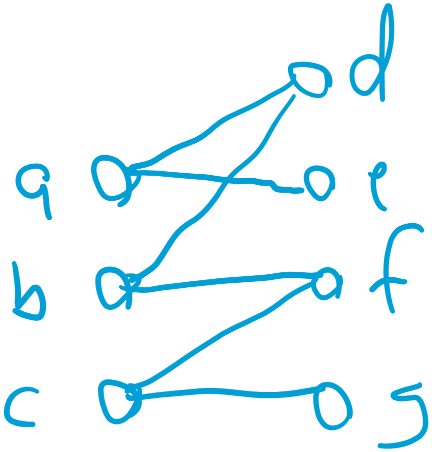
Even better fixed-parameter algorithms for bicluster editing

Manuel Lafond

UNIVERSITÉ DE SHERBROOKE

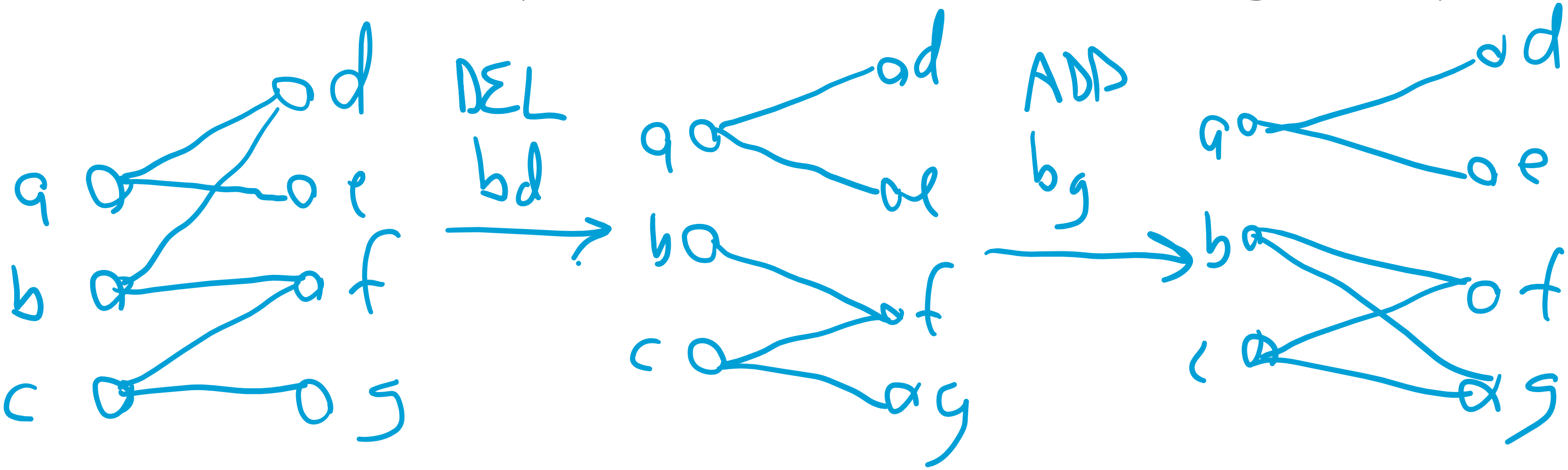
Bicluster editing

- **<u>Given</u>**: bipartite graph $G = (V_1 \cup V_2, E)$
- **<u>Goal</u>** : add/remove a minimum number of edges so that every connected component is a bicluster (a complete bipartite graph).



Bicluster editing

- **<u>Given</u>**: bipartite graph $G = (V_1 \cup V_2, E)$
- **<u>Goal</u>** : add/remove a minimum number of edges so that every connected component is a bicluster (a complete bipartite graph).



Some applications

- Gene expression analysis
- Social network analysis
- Phylogenetic analysis

Previous work

- NP-hard on subcubic graphs [Drange & al, 2015]
  - (actually ETH-hard)
- $O^*(4^k)$ time FPT algorithm [Protti & al., 2006]

Previous work

- NP-hard on subcubic graphs [Drange & al, 2015]
  - (actually ETH-hard)
- $O^*(4^k)$ time FPT algorithm [Protti & al., 2006]
  - Kernel of $4k^2 + 6k$ vertices

Previous work

- NP-hard on subcubic graphs [Drange & al, 2015]
  - (actually ETH-hard)
- $O^*(4^k)$ time FPT algorithm [Protti & al., 2006]
  - Kernel of $4k^2 + 6k$ vertices
- $O^*(3.24^k)$ time FPT algorithm [Guo & al., 2008]

Previous work

- NP-hard on subcubic graphs [Drange & al, 2015]
  - (actually ETH-hard)
- $O^*(4^k)$ time FPT algorithm [Protti & al., 2006]
  - Kernel of $4k^2 + 6k$ vertices
- $O^*(3.24^k)$ time FPT algorithm [Guo & al., 2008]
  - Kernel of $4k$ vertices (inaccuracy)
  - More likely $6k$, but remains to be proved

In this work

- $O^*(2.695^k)$ time FPT algorithm
  - Simple branching algorithm
- Kernel of $5k$ vertices

$O^*(2.695^k)$ time algorithm

- Focus on u, v on the same side that are conflicting:
  - $N(u) \cap N(v) \neq \emptyset$
  - $N(u) \neq N(v)$

$O^*(2.695^k)$ time algorithm

- Focus on u, v on the same side that are conflicting:
  - $N(u) \cap N(v) \neq \emptyset$
  - $N(u) \neq N(v)$
- Either u, v in the same bicluster, or not.
  - Both choices induce some cost.

$O^*(2.695^k)$ time algorithm

- Put u, v in same bicluster
  - Each node in $N(u) \Delta N(v)$ requires an insertion or deletion.
  - $2^{|N(u) \Delta N(v)|}$ ways, each requiring $|N(u) \Delta N(v)|$ edits.

$O^*(2.695^k)$ time algorithm

- Put u, v in different bicluster
  - Each node in $N(u) \cap N(v)$ requires a deletion.
  - $2^{|N(u) \cap N(v)|}$ ways, each requiring $|N(u) \cap N(v)|$ edits.

```
function biclusterize(G, k)
    if k < 0 then  Report "NO" and return
    Remove from G all bicluster connected components (Rule 1)
    if G has no vertex then  Report "YES" and return
    if G has maximum degree 2 then  Solve G in polynomial time

    Let u, v ∈ V(G) such that N(u) ∩ N(v) ≠ ∅ and N(u)△N(v) ≠ ∅
    Let R_u be the twin class of u and R_v be the twin class of v
    /*Put u, v in the same bicluster*/
    for each subset Z of N(u)△N(v) do
        Obtain G' from G by:
         inserting all missing edges between R_u ∪ R_v and Z and
         deleting all edges between R_u ∪ R_v and (N(u)△N(v)) \ Z
        Let h be the number of edges modified from G to G'
        biclusterize(G', k − h)
    end
    /*Put u, v in different biclusters*/
    for each subset Z of N(u) ∩ N(v) do
        Obtain G' from G by:
         deleting all edges between R_u and (N(u) ∩ N(v)) \ Z and
         deleting all edges between R_v and Z
        Let h be the number of edges modified from G to G'
        biclusterize(G', k − h)
    end
    if some recursive call reported "YES" then report "YES"
    else report "NO"
```

- Let $c = N(u) \cap N(v)$ and $d = N(u) \Delta N(v)$
- Branching recurrence

$$f(k) = 2^c f(k - c) + 2^d f(k - d)$$

- Let $c = N(u) \cap N(v)$ and $d = N(u) \Delta N(v)$
- Branching recurrence

$$f(k) = 2^c f(k - c) + 2^d f(k - d)$$

- **Lemma**: for fixed c, the lower d is, the worse the complexity (and vice-versa).

- Let $c = N(u) \cap N(v)$ and $d = N(u) \Delta N(v)$
- Branching recurrence

$$f(k) = 2^c f(k - c) + 2^d f(k - d)$$

- **Lemma**: for fixed c, the lower d is, the worse the complexity (and vice-versa).

- **Lemma**: worst case occurs when $c = d = 1$.

- **Lemma**: worst case occurs when $c = d = 1$.
- This worst case occurs only if all conflicting pairs u, v have degrees 1 and 2.
  - If so, problemn is easy (handle cycles and paths)

- **Lemma**: worst case occurs when $c = d = 1$.
- This worst case occurs only if all conflicting pairs u, v have degrees 1 and 2.
  - If so, problemn is easy (handle cycles and paths)
- We may assume that there is a conflicting pair u, v in which deg(u) $\geq$ 3.
  - Worst case is $c = 2, d = 1$ or $c = 1, d = 2$

- **Lemma**: worst case occurs when $c = d = 1$.
- This worst case occurs only if all conflicting pairs u, v have degrees 1 and 2.
  - If so, problemn is easy (handle cycles and paths)
- We may assume that there is a conflicting pair u, v in which deg(u) $\geq$ 3.
  - Worst case is $c = 2, d = 1$ or $c = 1, d = 2$
- Worst case recurrence:

$$f(k) = 2^1 f(k - 1) + 2^2 f(k - 2)$$

- **<u>Lemma</u>**: worst case occurs when $c = d = 1$.
- This worst case occurs only if all conflicting pairs u, v have degrees 1 and 2.
  - If so, problemn is easy (handle cycles and paths)
- We may assume that there is a conflicting pair u, v in which deg(u) $\geq$ 3.
  - Worst case is $c = 2, d = 1$ or $c = 1, d = 2$

- Worst case recurrence:

$$f(k) = 2^1 f(k - 1) + 2^2 f(k - 2)$$

- branching factor 3.237 => $O^*(3.237^k)$ time algorithm

- $O^*(3.237^k)$ time algorithm
- Actually the same as in [Guo & al., 2008]

- **Theorem**: if G has a vertex of degree 1, then one can attain branching factor 2.066.

- **Theorem**: if G has **no** vertex of degree 1, then the branching factor of the main algorithm is 2.695.
  - Assuming it uses degree 1 special branching if a degree 1 vertex apears in a recursion.

- **Theorem**: if G has **no** vertex of degree 1, then the branching factor of the main algorithm is 2.695.
  - Assuming it uses degree 1 special branching if a degree 1 vertex apears in a recursion.

- => $O^*(2.695^k)$ time algorithm

Theorem: BICLUSTER EDITING admits a kernel of size 5k.

Kernel of size $5k$

- **Rule 1** : if a connected component X of G is a bicluster, remove X from G.

Kernel of size $5k$

- **Rule 2** : if there is a set of twins R such that $|R| > |N\big(N(R)\big)\backslash R|$, remove any vertex from R.
    - twins = on same side, have same neighbors

Kernel of size $5k$

- **Rule 1** : if a connected component X of G is a bicluster, remove X from G.
- **Rule 2** : if there is a set of twins R such that $|R| > |N(N(R)) \setminus R|$, remove any vertex from R.
  - twins = on same side, have same neighbors
- Rules from [Guo & al., 2008]
  - Claim : imply $4k$ kernel

Kernel of size $5k$

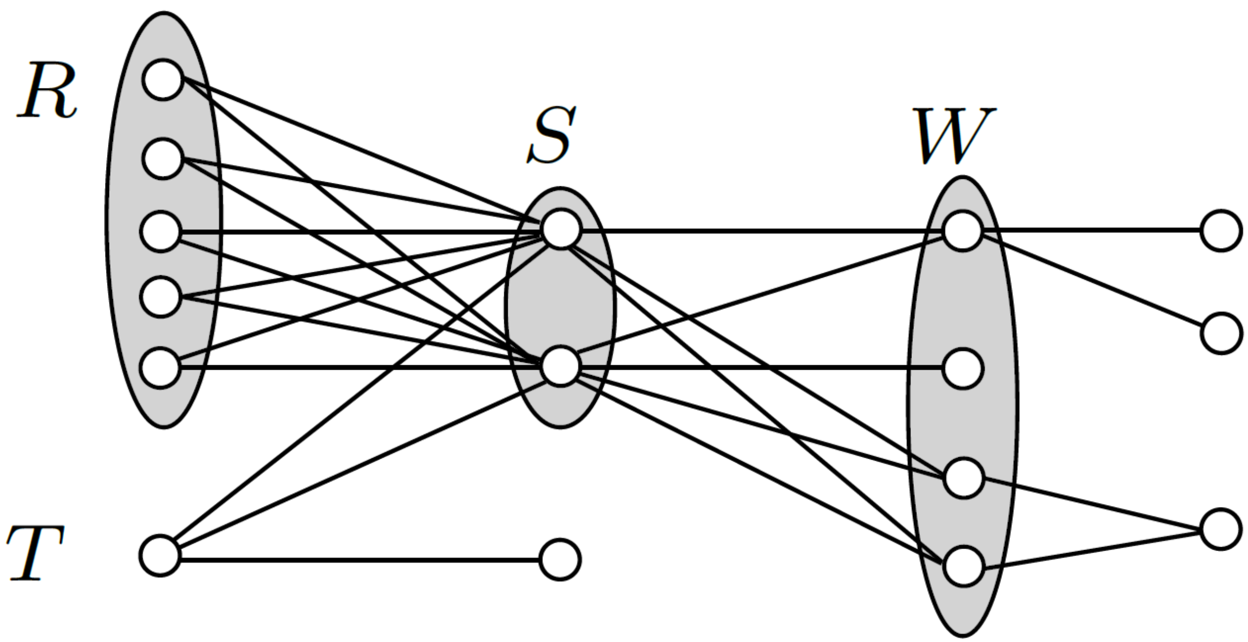
- **Rule 1** : if a connected component X of G is a bicluster, remove X from G.
- **Rule 2** : if there is a set of twins R such that $|R| > |N(N(R))\backslash R|$, remove any vertex from R.
  - twins = on same side, have same neighbors
- Rules from [Guo & al., 2008]
  - Claim : imply $4k$ kernel
  - False : P_6 cannot be reduced

Kernel of size $5k$

- **Defn** : let R be twins. A vertex t is a sister of R if $N(t) = N(R) \cup \{u\}$ for some u, and t has no twins.

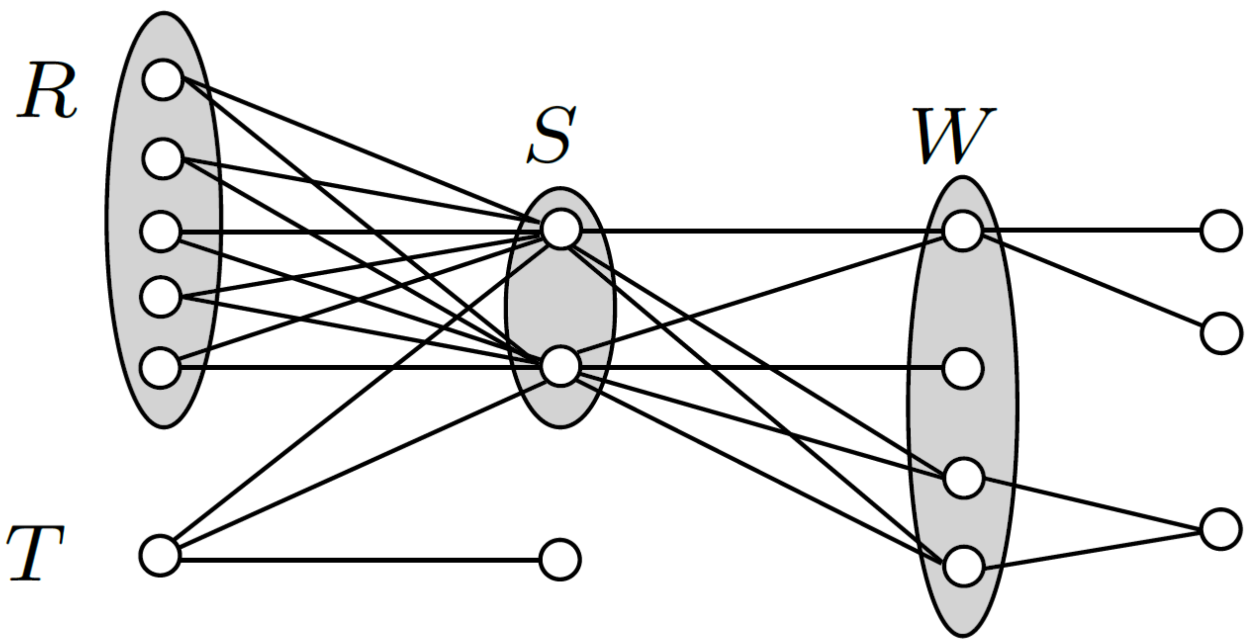
Kernel of size $5k$

- **Defn** : let R be twins. A vertex t is a sister of R if $N(t) = N(R) \cup \{u\}$ for some u, and t has no twins.



Kernel of size $5k$

- **Defn** : let R be twins. A vertex t is a sister of R if $N(t) = N(R) \cup \{u\}$ for some u, and t has no twins.
- **Rule 3** : let R be twins and T its sisters.
  If $|R| > |N(N(R)) \setminus \{R \cup T\}|$, then remove any edge from $t \in T$ to its neighbor outside $N(R)$.



Kernel of size $5k$

- **Thm**: Rules 1,2,3 are safe and lead to a kernel of size $5k$.
- Read paper for proof idea.

- Open: true kernel size of Rules 1,2,3?

Conclusion

- Better analysis = better than $O^*(2.695^k)$?
- Better analysis = better than $5k$ kernel?

- Ideas applicable to Cluster editing?
- Existence of $O^*(2^k)$ algorithm or better?