Parameterized Complexity of

Maximum Degree Contraction

IPEC 2020

Saket Saurabh

The Institute Of Mathematical Sciences, HBNI, Chennai, India

University of Bergen, Bergen, Norway

Prafullkumar Tale

CISPA Helmholtz Center for Information Security, Saarbrucken, Germany

Short Version

Maximum Degree Contraction

Input: Graph G, ints k, d

Question: Can we contract at most k edges in G s.t. the maximum degree in the resulting graph is at most d?

Para: k + d

Parameterized complexity of three edge contraction problems with degree constraints. — Belmonte et. al. *Acta Informatica*, (2014).

Thm 1: Algo running in time (k + d)^O(k).

Thm 2: When d = 2, linear kernel => 2^O(k) algo.

Q: Poly kernel k + d?

R1: No algo running in time $n^{o(k)}$ unless ETH fails.

No algo running in time $(k + d)^{o(k)}$ unless ETH fails.

Thm 1: Algo running in time $(k + d)^{O(k)}$.

(Optimum when k, d both part of input)

R2: Algo running in time $2^{O(kd)}$.

Thm 2: When d = 2, linear kernel => $2^{O(k)}$ algo.

Q: Poly kernel k + d? R3: No, unless NP $\subseteq$ coNP/poly

R1: No algo running in time $n^{o(k)}$ unless ETH fails.

(k x k)-MultiColored-Clique

R2: Algo running in time $2^{O(kd)}$.

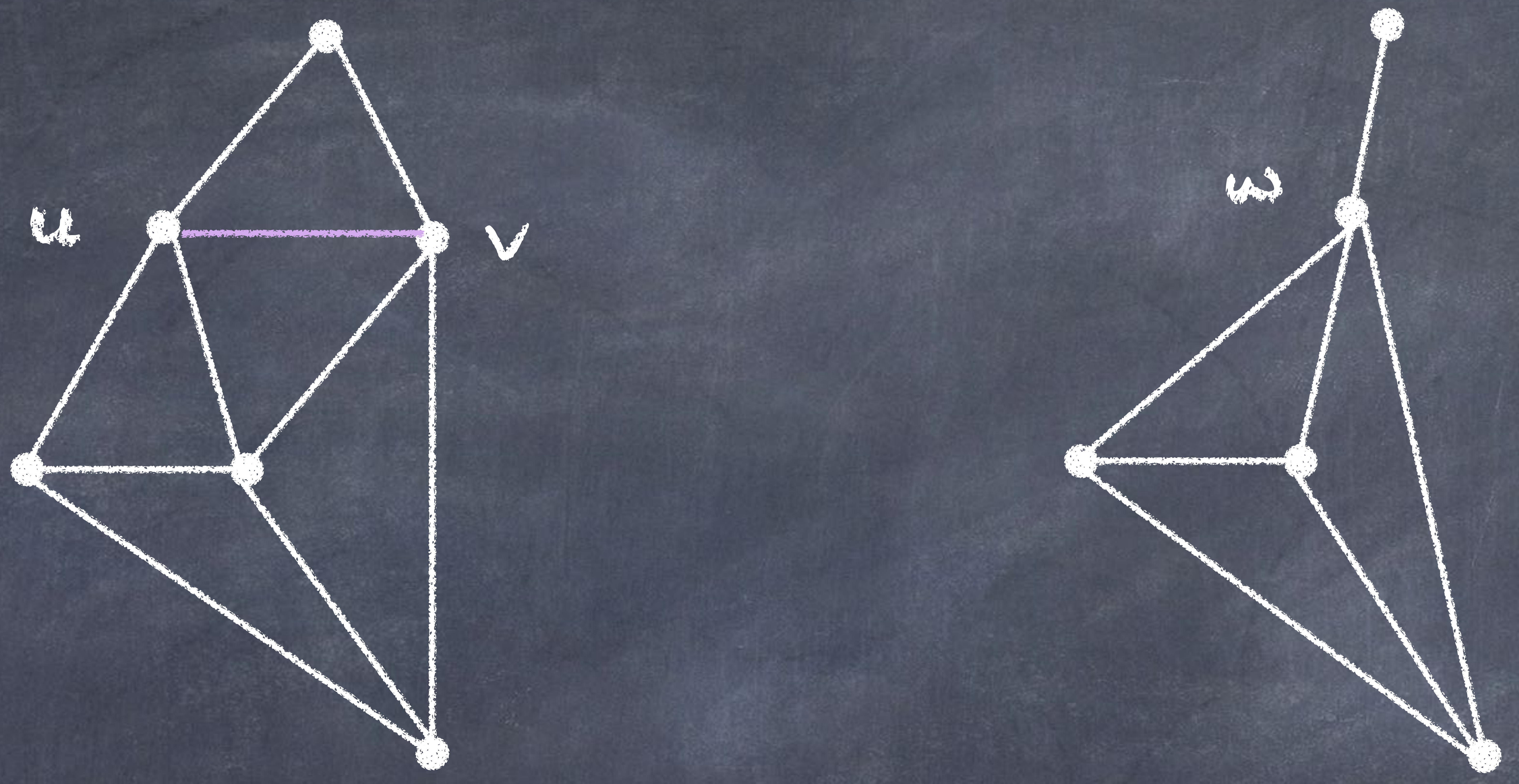
Universal Sets + Branching

Q: Poly kernel k + d?

R3: No, unless NP ⊆ coNP/poly

RBDS para |B| + log |R|

Longer Version



Edge contraction

No self-loop
No parallel edges

Known Results

Our Contributions

FPT Algorithm

Open Questions

$\mathcal{G}$ – Contraction

Input: Graph G, int k

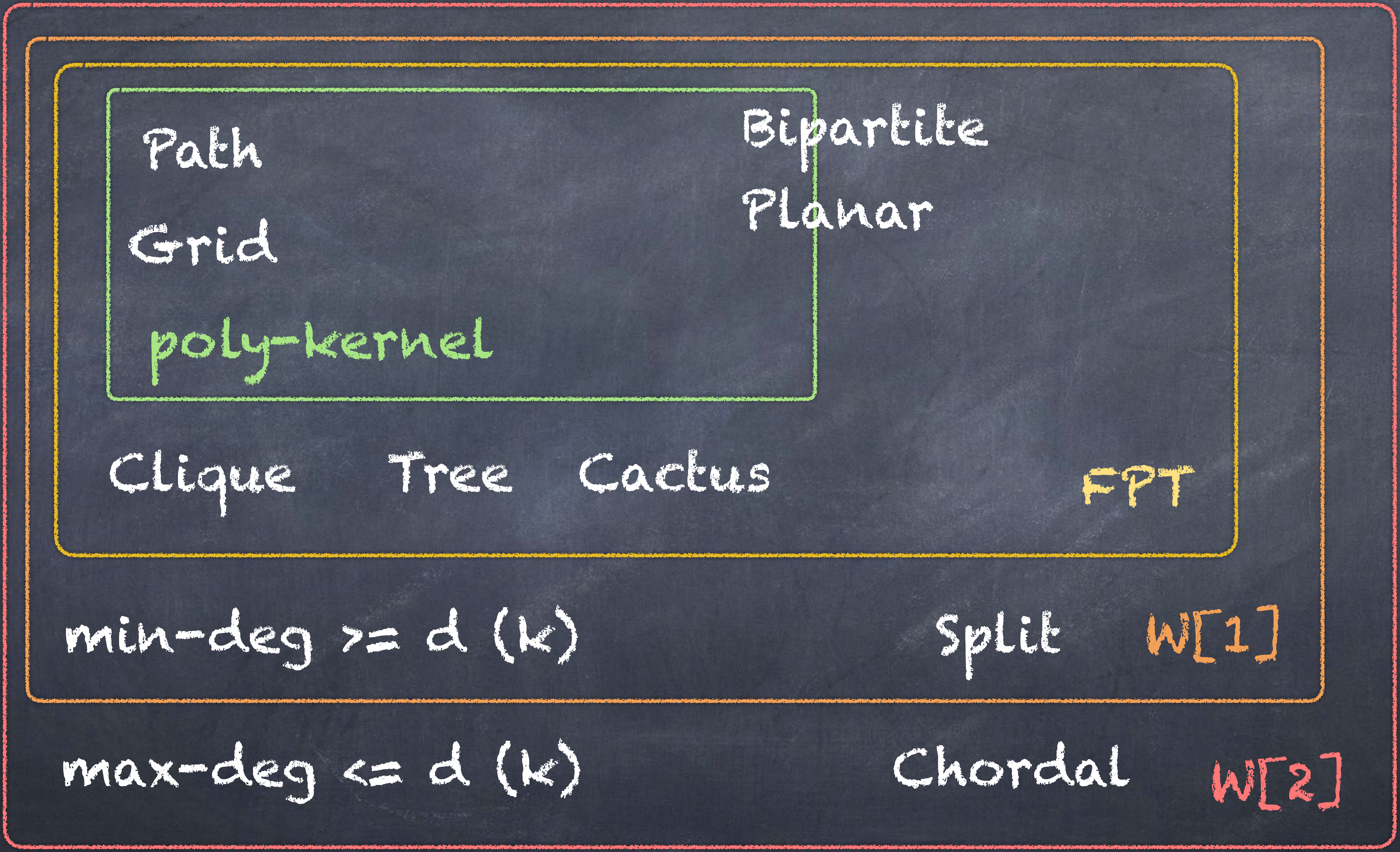
Question: Can we contract at most k edges in G s.t. resulting graph is in $\mathcal{G}$?

Para: k

Path Tree Cactus Planar Bipartite Grid

Clique Split Chordal

max-deg <= d (k) min-deg >= d (k)


Path
Bipartite
Planar
Grid
poly-kernel
Clique
Tree
Cactus
FPT
min-deg >= d (k)
Split
W[1]
max-deg <= d (k)
Chordal
W[2]

Known Results

Our Contributions

FPT Algorithm

Open Questions

Theorem 1. *Unless ETH fails, there is no algorithm that given any instance* (G, k, d) *of* Maximum Degree Contraction *runs in time* $n^{o(k)}$ *and correctly determines whether it is a* Yes *instance.*

Brute force algo: n^O(k)

Theorem 2. *There is an algorithm that given an instance* (G, k, d) *of* Maximum Degree Contraction *runs in time* $2^{O(dk)} \cdot n^{O(1)}$ *and correctly determines whether it is a* Yes *instance.*

Lower bound: 2^o(kd)

Theorem 3. *Unless* NP ⊆ coNP/poly, Maximum Degree Contraction, *parameterized by* k + d, *does not admit a polynomial compression.*

Known Results

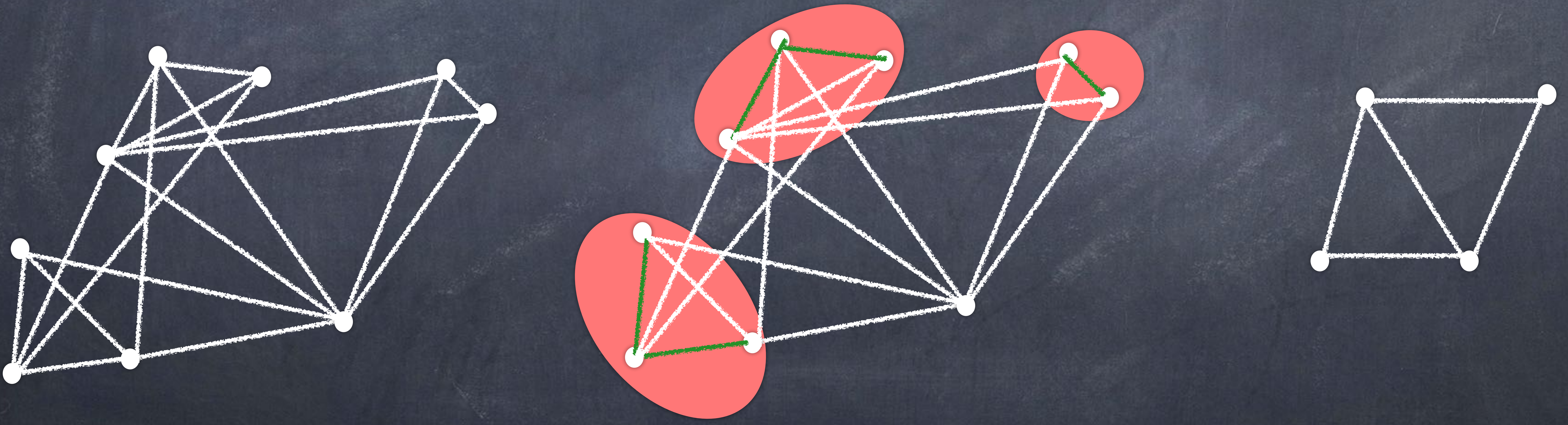
Our Contributions

FPT Algorithm

Open Questions

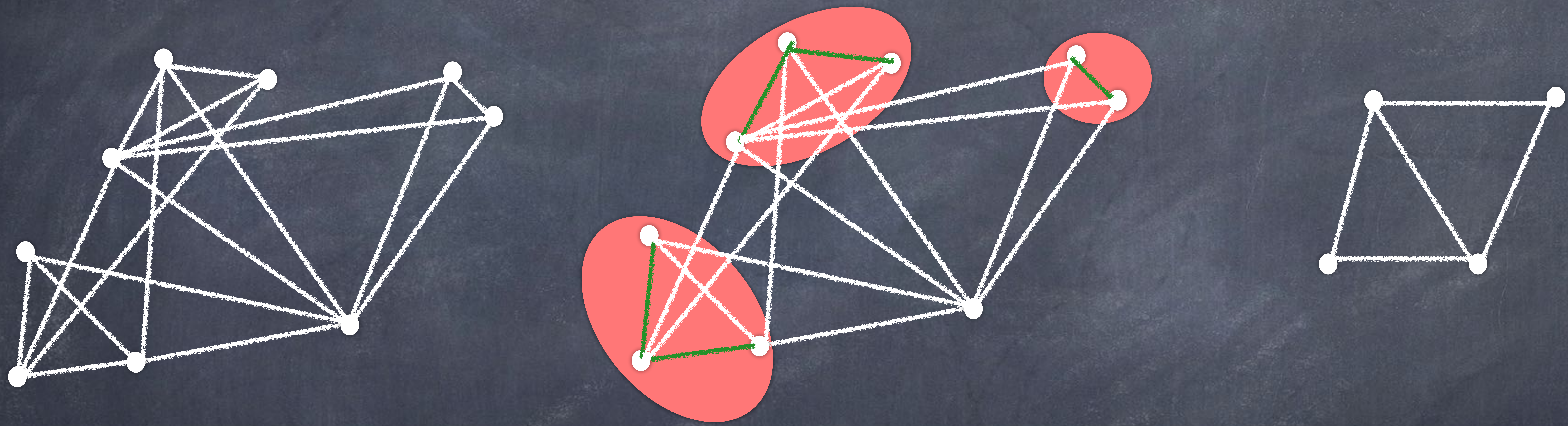
Max-Deg Contra:: Partition of V(G) s.t.

(a) Each part (called witness set) is connected
(b) Any witness set is adj with at most d other



Max-Deg Contra:: Partition of V(G) s.t.

(a) Each part (called witness set) is connected
(b) Any witness set is adj with at most d other



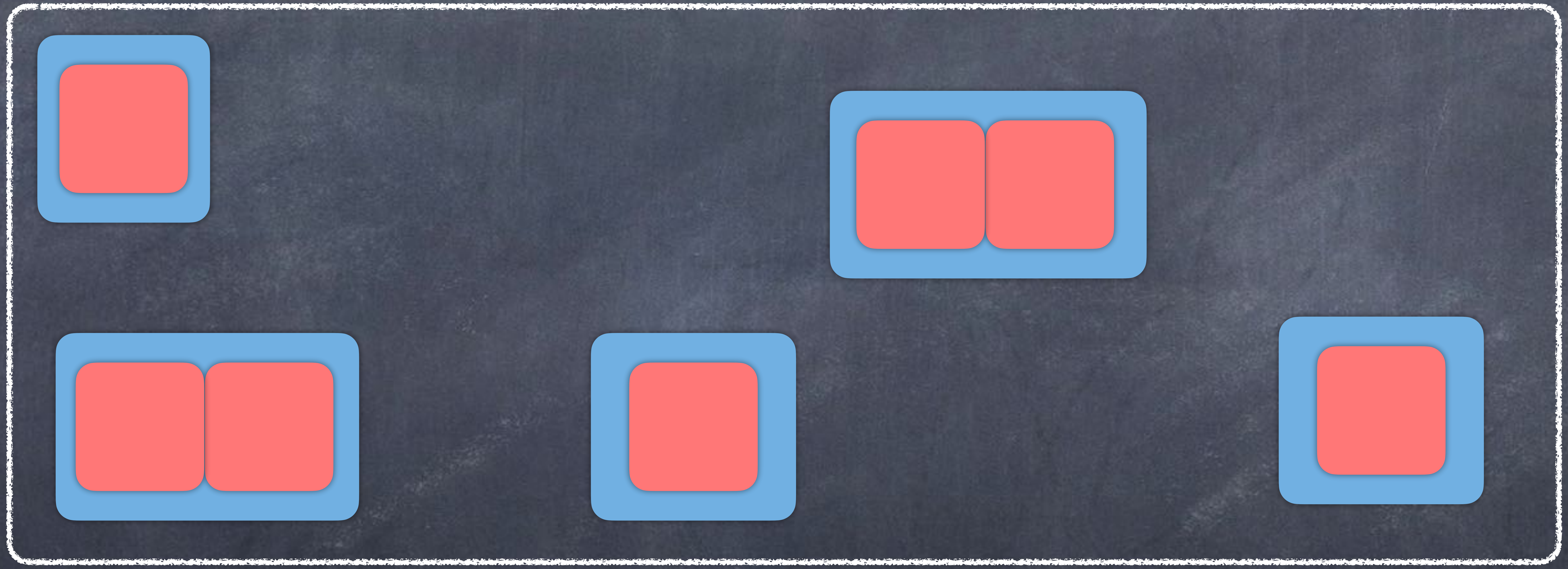
(a) + Solution size => 2k vert in big witness sets

(b) + Solution size => kd vert adj to big witness sets

A 2-coloring of V(G) that colors

2k vert in big witness sets with red-color

kd vert adj to big witness sets with blue-color



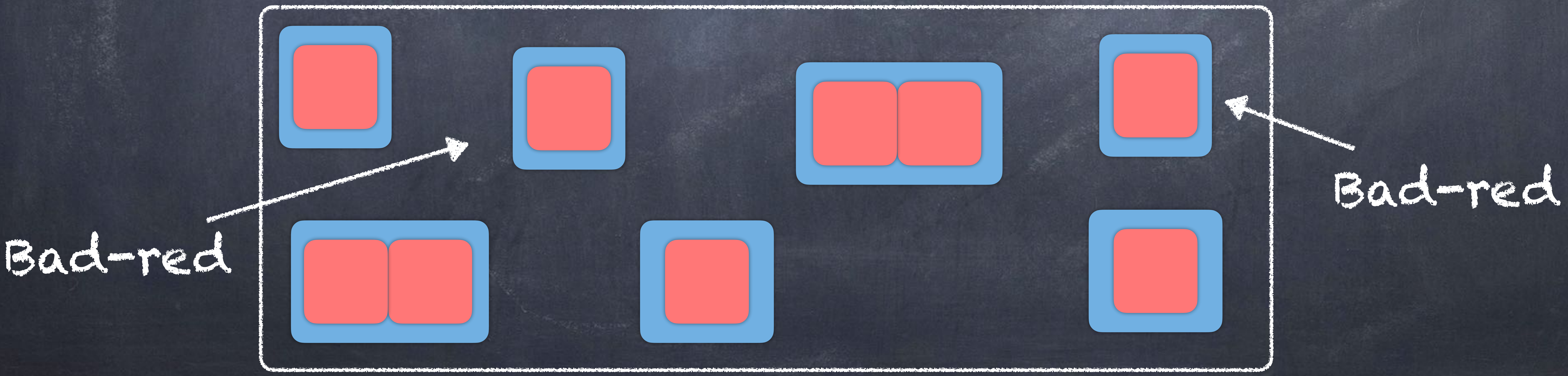
(n, 2k + dk) Universal Sets

(n, 2k + dk) Universal Sets

- Collection of exp(kd) n log n 2-colorings of [n]
- For any set X of size 2k + dk, and its partition (X1, X2), there is a coloring 'compatible' with it.

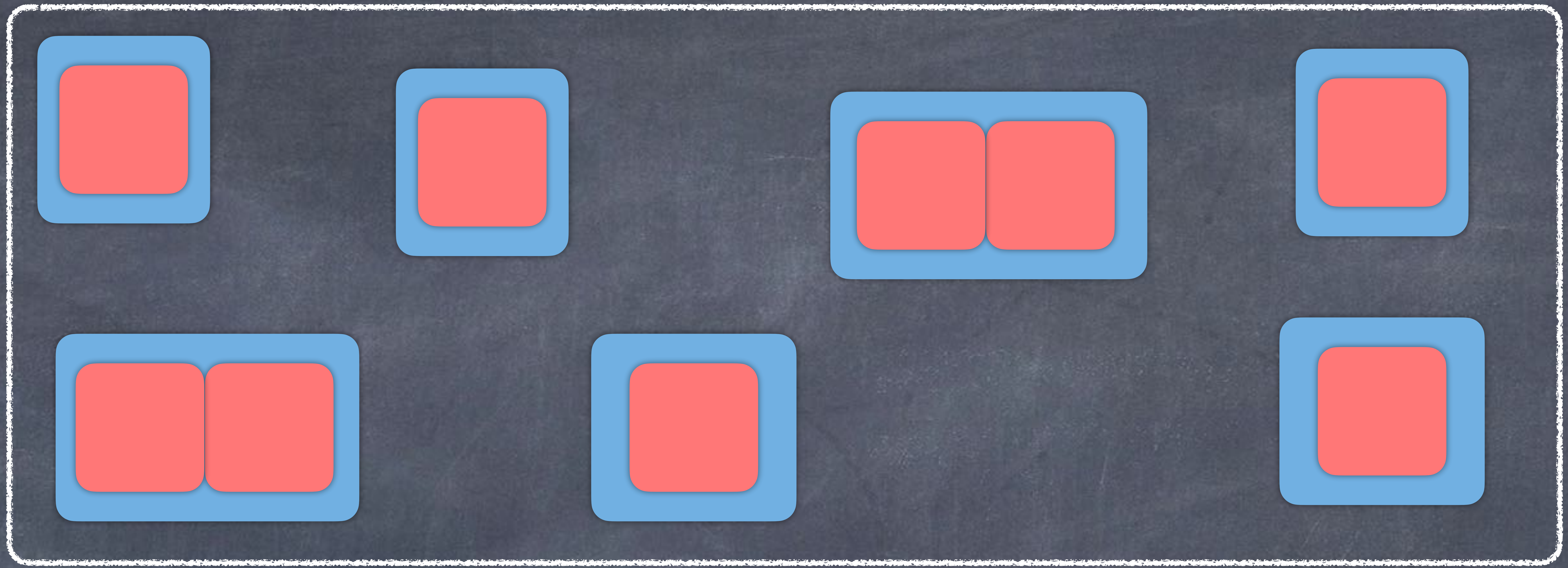
X1 - vert in big witness sets

X2 - vert adj to big witness sets



Bad-red

(Does not contain a big witness set)



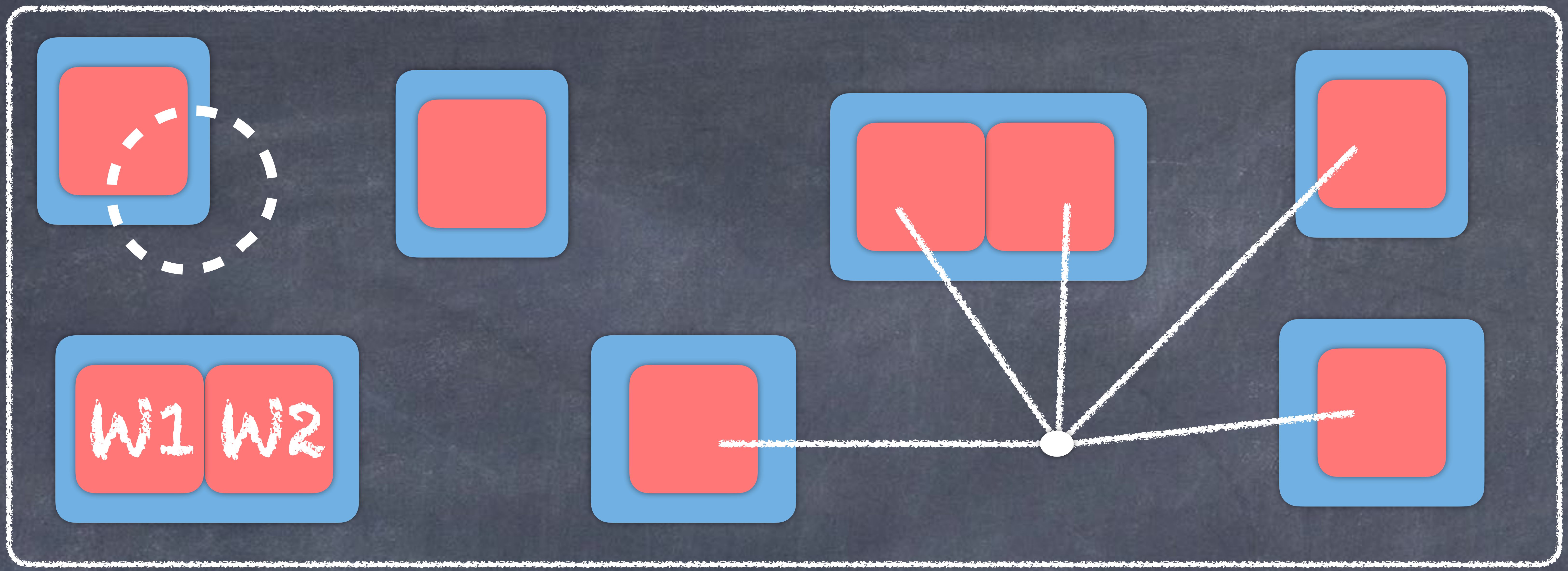
Maximum Degree Contraction <=>

Given a 'compatible' 2-coloring: good-red vs bad-red

Fix a hypothetical solution F. X1 = V(F), X2 = N(V(F))

|F| <= k => |X1| <= 2k; |X2| <= kd

One of 2-colouring in (n, 2k + dk)-Universal Sets is compatible with (X1, X2).



Given a 'compatible' 2-coloring: good-red vs bad-red

- Any witness set is completely in red-part
- Any red-part is either union of witness sets or doesn't intersect any witness set
- Any vertex in blue-part can see at most d red-parts

- Any red-part is either union of witness sets or doesn't intersect any witness set
- Any vertex in blue-part can see at most d red-parts

```
while( deg(u) >= d + 1 )
   if u in red-part: contract that part
   if u in blue-part: branch over 2^d possibilities
```

Measure (k) drops in each case

Running time = |Universal Sets| * |Branching|
= exp(kd) * exp(kd)

Known Results

Our Contributions

FPT Algorithm

Open Questions

Q1: When $d = 3$, does MAXIMUM DEGREE CONTRACTION admits a poly?

- $d = 2$, kernel of size $O(k)$ on connected graph
- arbitrary d, no poly kernel

Q2: Lossy kernel for MAXIMUM DEGREE CONTRACTION?

Thank you!