Dual Parametrization of Weighted Coloring

Júlio Araújo

joint work with
Victor Campos, Carlos Vinícius Lima, Vinícius dos Santos,
Ignasi Sau & Ana Silva

17 de setembro de 2020

Weighted Coloring

We are given a graph G together with a weight function $w : V(G) \to \mathbb{R}^+$.

A (proper) k-coloring of G is a partition $c = (S_i)_{i \in [1,k]}$ of $V(G)$ into k stable sets $S_1, \ldots, S_k$.

Weighted Coloring

We are given a graph G together with a weight function $w : V(G) \to \mathbb{R}^+$.

A (proper) k-coloring of G is a partition $c = (S_i)_{i \in [1,k]}$ of $V(G)$ into k stable sets $S_1, \ldots, S_k$.

The weight of a color S_i is $w(i) = \max_{v \in S_i} w(v)$.

Weighted Coloring

We are given a graph G together with a weight function $w : V(G) \to \mathbb{R}^+$.

A (proper) k-coloring of G is a partition $c = (S_i)_{i \in [1,k]}$ of $V(G)$ into k stable sets $S_1, \ldots, S_k$.

The weight of a color S_i is $w(i) = \max_{v \in S_i} w(v)$.

The weight of a coloring c is $w(c) = \sum_{i=1}^{k} w(i)$.

Weighted Coloring

We are given a graph G together with a weight function $w : V(G) \to \mathbb{R}^+$.

A (proper) k-coloring of G is a partition $c = (S_i)_{i \in [1,k]}$ of $V(G)$ into k stable sets $S_1, \ldots, S_k$.

The weight of a color S_i is $w(i) = \max_{v \in S_i} w(v)$.

The weight of a coloring c is $w(c) = \sum_{i=1}^{k} w(i)$.

The weighted chromatic number of a pair (G, w) is

$$\sigma(G, w) \;=\; \min\{w(c) \mid c \text{ is a proper coloring of } G\}.$$

Weighted Coloring

We are given a graph G together with a weight function $w : V(G) \to \mathbb{R}^+$.

A (proper) k-coloring of G is a partition $c = (S_i)_{i \in [1,k]}$ of $V(G)$ into k stable sets $S_1, \ldots, S_k$.

The weight of a color S_i is $w(i) = \max_{v \in S_i} w(v)$.

The weight of a coloring c is $w(c) = \sum_{i=1}^{k} w(i)$.

The weighted chromatic number of a pair (G, w) is

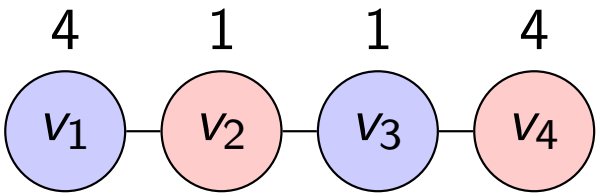
$$\sigma(G, w) \;=\; \min\{w(c) \mid c \text{ is a proper coloring of } G\}.$$

For a positive integer r, we define

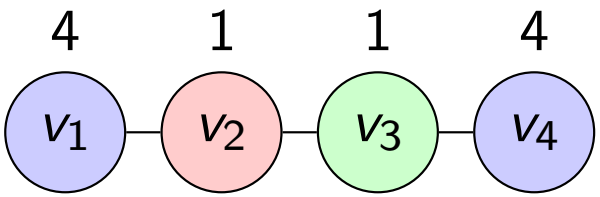
$$\sigma(G, w; r) \;=\; \min\{w(c) \mid c \text{ is a proper } r\text{-coloring of } G\}.$$

Some authors call it Max Coloring!

Example



$$w(c) = w(S_1) + w(S_2) = 8$$



$$w(c) = w(S_1) + w(S_2) + w(S_3) = 6$$

$$\sigma(P_4, w) = 6$$

$$\sigma(P_4, w; 2) = 8$$

Remark

If $G = (V, E, w)$ has all the weights equal to 1:

- $\sigma(G, w) = \chi(G)$;

Remark

If $G = (V, E, w)$ has all the weights equal to 1:

- $\sigma(G, w) = \chi(G)$;
- Hardness results hold for $\sigma(G, w)$.

Some complexity results up to 2006

Polynomial

- Bipartite graphs with 2 different weights [Demange et al. 2002]
- P_5-free bipartite graphs [Demange et al. 2005]
- P_4-**free graphs** (i.e., Cographs) [Demange et al. 2002]

NP-complete

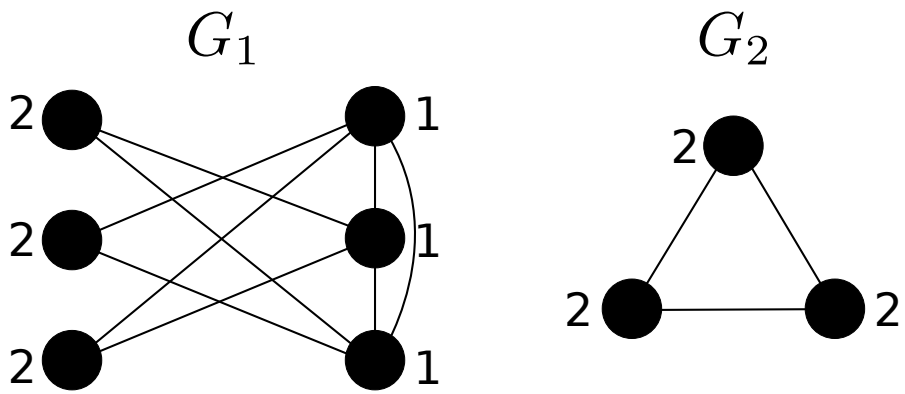
- Interval graphs [Escoffier et al. 2005]
- Bipartite graphs with $\Delta(G) \leq 14$ [Demange et al. 2002]
- Triangle-free planar graphs with $\Delta(G) \leq 4$ [Demange et al. 2005]
- Split graphs [Demange et al. 2002]
- $\Rightarrow$ P_5-**free graphs**

Approximation algorithms, etc.

P_4-sparse graph

$G = (V, E)$ is P_4-sparse if $\forall H \subseteq V$ s. t. $|H| \leq 5$ induces at most one P_4.

Problem with the disjoint union! $\rightarrow$ an optimal coloring of $G = G_1 \cup G_2$ is not given by optimal colorings of G_1 and G_2.



P_4-sparse graph

$G = (V, E)$ is P_4-sparse if $\forall H \subseteq V$ s. t. $|H| \leq 5$ induces at most one P_4.

Theorem (A., Linhares Sales, Sau, 2007)

$\sigma(G, w)$ can be computed in poly time for a subclass of P_4-sparse graphs.

Theorem (A., Linhares Sales, Sau, 2007)

There is a 2-approx. algorithm to $\sigma(G, w)$, for P_4-sparse graphs G.

Hajós-like Theorem

Theorem (A., Linhares Sales, 2007)

$\sigma(G, w) \geq k$ if, and only if, G contains a weighted k-constructible subgraph.

Open problem | Is Weighted Coloring **polynomial** on trees/forests?
More generally, on graphs of bounded treewidth?

Open problem Is Weighted Coloring **polynomial** on trees/forests?
More generally, on graphs of bounded treewidth?

Some partial results:

- PTAS on bounded treewidth graphs. [Escoffier, Monnot, Paschos. 2006]
- Polynomial on the class of trees where vertices with degree at least three induce a stable set. [Kavitha, Mestre. 2012]
  - Some improvements for subclasses of trees in [Benkoczi, Dahal, Gaur, 2016]

Complexity of weighted coloring on trees (or forests)

On an n-vertex graph of treewidth t, $\sigma(G, w; r)$ can be computed in time

$$n^{O(r)} \cdot r^{O(t)}.$$ [Guan, Zhu. 1997]

Complexity of weighted coloring on trees (or forests)

On an n-vertex graph of treewidth t, $\sigma(G, w; r)$ can be computed in time

$$n^{O(r)} \cdot r^{O(t)}. \qquad \text{[Guan, Zhu. 1997]}$$

Largest r s. t. $\sigma(G, w; r) = \sigma(G, w)$ satisfies $r \leq \chi_{\mathsf{FF}}(G)$. [Guan, Zhu. 1997]

Complexity of weighted coloring on trees (or forests)

On an n-vertex graph of treewidth t, $\sigma(G, w; r)$ can be computed in time

$$n^{O(r)} \cdot r^{O(t)}. \qquad \text{[Guan, Zhu. 1997]}$$

Largest r s. t. $\sigma(G, w; r) = \sigma(G, w)$ satisfies $r \leq \chi_{\mathrm{FF}}(G)$. [Guan, Zhu. 1997]

For any graph G, it holds that $\chi_{\mathrm{FF}}(G) = O(t \log n)$. [Linhares, Reed. 2006]

Complexity of weighted coloring on trees (or forests)

On an n-vertex graph of treewidth t, $\sigma(G, w; r)$ can be computed in time

$$n^{O(r)} \cdot r^{O(t)}. \qquad \text{[Guan, Zhu. 1997]}$$

Largest r s. t. $\sigma(G, w; r) = \sigma(G, w)$ satisfies $r \leq \chi_{\mathrm{FF}}(G)$. [Guan, Zhu. 1997]

For any graph G, it holds that $\chi_{\mathrm{FF}}(G) = O(t \log n)$. [Linhares, Reed. 2006]

$\Longrightarrow$ Weighted Coloring can be solved on forests in time $n^{O(\log n)} = 2^{O(\log^2 n)}$ (quasi-polynomial).

Theorem (A., Nisse, Pérennes. 2014)

Unless the ETH fails, there is no algorithm computing the weighted chromatic number of n-vertex trees in time $n^{o(\log n)}$.

Theorem (A., Nisse, Pérennes. 2014)

Unless the ETH fails, there is no algorithm computing the weighted chromatic number of n-vertex trees in time $n^{o(\log n)}$.

Exponential Time Hypothesis (ETH): the 3-SAT problem on formulas with n variables cannot be solved in subexponential time, that is, $2^{o(n)}$.

[Impagliazzo, Paturi, Zane. 2001]

Theorem (A., Nisse, Pérennes. 2014)

Unless the ETH fails, there is no algorithm computing the weighted chromatic number of n-vertex trees in time $n^{o(\log n)}$.

Exponential Time Hypothesis (ETH): the 3-SAT problem on formulas with n variables cannot be solved in subexponential time, that is, $2^{o(n)}$.

[Impagliazzo, Paturi, Zane. 2001]

That is, the running time $n^{O(\log n)}$ is tight under the ETH.

Can we relax the complexity hypothesis?

Can we relax the complexity hypothesis?

Providing hardness results for computing $\sigma(G, w)$ and $\sigma(G, w; r)$ when G is a forest, relying on complexity assumptions weaker than the ETH.

Can we relax the complexity hypothesis?

Providing hardness results for computing $\sigma(G, w)$ and $\sigma(G, w; r)$ when G is a forest, relying on complexity assumptions weaker than the ETH.

We study the problem from the viewpoint of parameterized complexity, and we assume the weaker hypothesis FPT $\neq$ W[1].

Can we relax the complexity hypothesis?

Providing hardness results for computing $\sigma(G, w)$ and $\sigma(G, w; r)$ when G is a forest, relying on complexity assumptions weaker than the ETH.

We study the problem from the viewpoint of parameterized complexity, and we assume the weaker hypothesis FPT $\neq$ W[1].

Indeed, it is well-known that

$$\boxed{\text{ETH}} \implies \boxed{\text{FPT} \neq \text{W[1]}} \implies \boxed{\text{P} \neq \text{NP}}$$

Can we relax the complexity hypothesis?

Providing hardness results for computing $\sigma(G, w)$ and $\sigma(G, w; r)$ when G is a forest, relying on complexity assumptions weaker than the ETH.

We study the problem from the viewpoint of parameterized complexity, and we assume the weaker hypothesis FPT $\neq$ W[1].

Indeed, it is well-known that

$$\boxed{\text{ETH}} \implies \boxed{\text{FPT} \neq \text{W[1]}} \implies \boxed{\text{P} \neq \text{NP}}$$

A few words on parameterized complexity

Instances of a parameterized problem: come with an integer parameter k.

A few words on parameterized complexity

Instances of a parameterized problem: come with an integer parameter k.

A parameterized problem is fixed-parameter tractable (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ solves the problem in time bounded by $f(k) \cdot |I|^c$.

A few words on parameterized complexity

Instances of a parameterized problem: come with an integer parameter k.

A parameterized problem is fixed-parameter tractable (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ solves the problem in time bounded by $f(k) \cdot |I|^c$.

W[1]-hard problems: any problem that admits a parameterized reduction from Independent Set parameterized by the size of the solution.

A few words on parameterized complexity

Instances of a parameterized problem: come with an integer parameter k.

A parameterized problem is fixed-parameter tractable (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ solves the problem in time bounded by $f(k) \cdot |I|^c$.

W[1]-hard problems: any problem that admits a parameterized reduction from Independent Set parameterized by the size of the solution.

W[2]-hard problems: any problem that admits a parameterized reduction from Dominating Set parameterized by the size of the solution.

A few words on parameterized complexity

Instances of a parameterized problem: come with an integer parameter k.

A parameterized problem is fixed-parameter tractable (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ solves the problem in time bounded by $f(k) \cdot |I|^c$.

W[1]-hard problems: any problem that admits a parameterized reduction from Independent Set parameterized by the size of the solution.

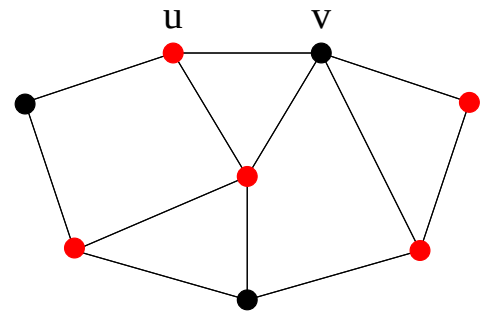
W[2]-hard problems: any problem that admits a parameterized reduction from Dominating Set parameterized by the size of the solution.

The theory of parameterized complexity is built based on FPT $\neq$ W[1].

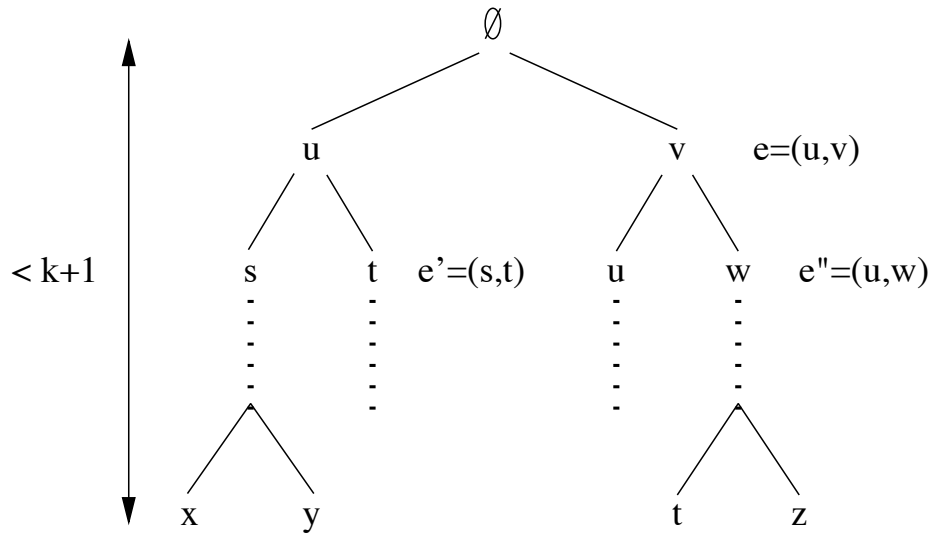
W[1]-hardness: strong evidence of not being FPT.
W[2]-hardness: even more!

Example of an FPT Algorithm

k-Vertex Cover: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \leq k$, covering $E(G)$?



Easy branching rule: Let (G, k) be an instance and let $e = \{u, v\} \in E(G)$. Branch into the two smaller instances $(G - u, k - 1)$ and $(G - v, k - 1)$.

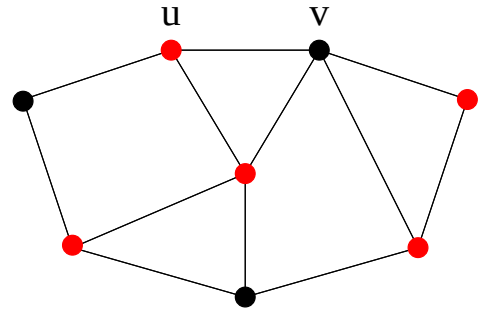


Running time: $O(2^k \cdot (m + n))$

Here, $n = |V(G)|$ and $m = |E(G)|$

Example of Kernelization Algorithm

k-Vertex Cover: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \leq k$, covering $E(G)$?



Reduction rule: over instance (G, k).

- If $\exists v$ s.t. $d(v) > k$, then (G, k) is YES iff $(G - v, k - 1)$ is YES.

$\Rightarrow$ we build an equivalent instance (G', k') with $\Delta(G') \leq k$!

If (G', k') has more than k^2 edges, the answer is NO.

Otherwise, G' has at most $2k^2$ vertices and at most k^2 edges.

G' is a kernel!

$\Rightarrow$ One can then make brute force on (G', k').

Running time: $O(2^{2k^2} \cdot (m + n))$

Theorem (A., Baste, Sau)

Given a weighted forest (G, w), computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G.

Theorem (A., Baste, Sau)

Given a weighted forest (G, w), computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G.

Consequences: W[1]-hard parameterized by treewidth, cliquewidth, maximum degree, maximum diameter of a connected component, number of colors, etc.

Theorem (A., Baste, Sau)

Given a weighted forest (G, w), computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G.

Consequences: W[1]-hard parameterized by treewidth, cliquewidth, maximum degree, maximum diameter of a connected component, number of colors, etc.

Theorem (A., Baste, Sau)

Given a weighted tree (G, w) and an integer r, computing $\sigma(G, w; r)$ is W[2]-hard parameterized by r.

Theorem (A., Baste, Sau)

Given a weighted forest (G, w), computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G.

Consequences: W[1]-hard parameterized by treewidth, cliquewidth, maximum degree, maximum diameter of a connected component, number of colors, etc.

Theorem (A., Baste, Sau)

Given a weighted tree (G, w) and an integer r, computing $\sigma(G, w; r)$ is W[2]-hard parameterized by r.

Note: results are incomparable to those of [A., Nisse, Pérennes. 2014]

Theorem (A., Baste, Sau)

Given a weighted forest (G, w), computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G.

Consequences: W[1]-hard parameterized by treewidth, cliquewidth, maximum degree, maximum diameter of a connected component, number of colors, etc.

Theorem (A., Baste, Sau)

Given a weighted tree (G, w) and an integer r, computing $\sigma(G, w; r)$ is W[2]-hard parameterized by r.

Note: results are incomparable to those of [A., Nisse, Pérennes. 2014]

Recall: on forests, $\sigma(G, w; r)$ can be computed in time $n^{O(r)}$.

Theorem (A., Baste, Sau)

Given a weighted forest (G, w), computing $\sigma(G, w)$ is W[1]-hard parameterized by the size of a largest connected component of G.

Consequences: W[1]-hard parameterized by treewidth, cliquewidth, maximum degree, maximum diameter of a connected component, number of colors, etc.

Theorem (A., Baste, Sau)

Given a weighted tree (G, w) and an integer r, computing $\sigma(G, w; r)$ is W[2]-hard parameterized by r.

Note: results are incomparable to those of [A., Nisse, Pérennes. 2014]

Recall: on forests, $\sigma(G, w; r)$ can be computed in time $n^{O(r)}$.

Corollary (A., Baste, Sau)

Assuming ETH, there is no algorithm that, given a weighted tree (G, w) and a positive integer r, computes $\sigma(G, w; r)$ in time $f(r) \cdot n^{o(r)}$ for any computable function f.

Dual parameterization: saving colors

Conclusion Weighted Coloring is a very hard problem!

Dual parameterization: saving colors

Conclusion Weighted Coloring is a very hard problem!

Let's try to give some good news:

Dual Weighted Coloring
Input: A vertex-weighted graph (G, w) and a positive integer k.
Parameter: k.
Output: Is $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$?

Dual parameterization: saving colors

Conclusion Weighted Coloring is a very hard problem!

Let's try to give some good news:

Dual Weighted Coloring
Input: A vertex-weighted graph (G, w) and a positive integer k.
Parameter: k.
Output: Is $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$?

(We assume that all vertex-weights are positive integers.)

Dual parameterization: saving colors

Conclusion Weighted Coloring is a very hard problem!

Let's try to give some good news:

Dual Weighted Coloring
Input: A vertex-weighted graph (G, w) and a positive integer k.
Parameter: k.
Output: Is $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k$?

(We assume that all vertex-weights are positive integers.)

The dual parameterization has proved to be useful for Vertex Coloring, Grundy Coloring, and b-Coloring.

[Chor, Fellows, Juedes. 2004]

[Havet, Sampaio. 2013]

Known results

Dual Weighted Coloring has been recently studied: [Escoffier. 2016]

Known results

Dual Weighted Coloring has been recently studied: [Escoffier. 2016]

Escoffier showed that the problem is FPT, namely:

- Algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.

Known results

Dual Weighted Coloring has been recently studied: [Escoffier. 2016]

Escoffier showed that the problem is FPT, namely:

- Algorithm running in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.
- Asked "what size of kernel can be achieved for the problem?"

Our results for Dual Weighted Coloring

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.
2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

   (Contrast with Dual Vertex Coloring, which admits a linear kernel.)

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

   (Contrast with Dual Vertex Coloring, which admits a linear kernel.)

3. Polynomial kernels on particular graph classes:

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

   (Contrast with Dual Vertex Coloring, which admits a linear kernel.)

3. Polynomial kernels on particular graph classes:

   - Quadratic kernel on comparability graphs.

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

   (Contrast with Dual Vertex Coloring, which admits a linear kernel.)

3. Polynomial kernels on particular graph classes:

   - Quadratic kernel on comparability graphs.
   - Cubic kernel on interval graphs.

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

   (Contrast with Dual Vertex Coloring, which admits a linear kernel.)

3. Polynomial kernels on particular graph classes:

   - Quadratic kernel on comparability graphs.
   - Cubic kernel on interval graphs.
   - Cubic kernel on subclass of circular-arc graphs.

Our results for Dual Weighted Coloring

1. FPT algorithm running in time $9^k \cdot n^{\mathcal{O}(1)}$.
   No algorithm in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ exists under the ETH.

2. Kernel with at most $(2^{k-1}+1)(k-1)$ vertices.

   No polynomial kernels unless NP $\subseteq$ coNP/poly,
   even on split graphs with only two different weights.

   (Contrast with Dual Vertex Coloring, which admits a linear kernel.)

3. Polynomial kernels on particular graph classes:

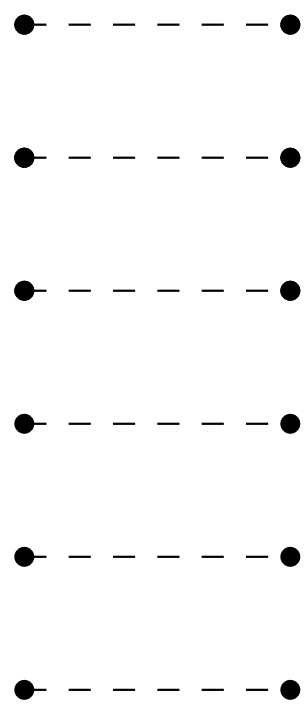
   - Quadratic kernel on comparability graphs.
   - Cubic kernel on interval graphs.
   - Cubic kernel on subclass of circular-arc graphs.
   - Kernel on Subclasses of split graphs, with lower bounds on the degrees.

An exponential kernel for Dual Weighted Coloring

We start with an idea already used, in particular, by [Escoffier. 2016]

An exponential kernel for Dual Weighted Coloring

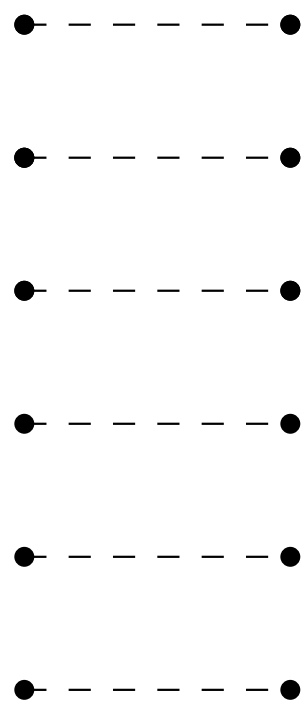
maximum unweighted antimatching $\overline{M}$



We compute (in poly time) a maximum unweighted antimatching $\overline{M}$.

An exponential kernel for Dual Weighted Coloring

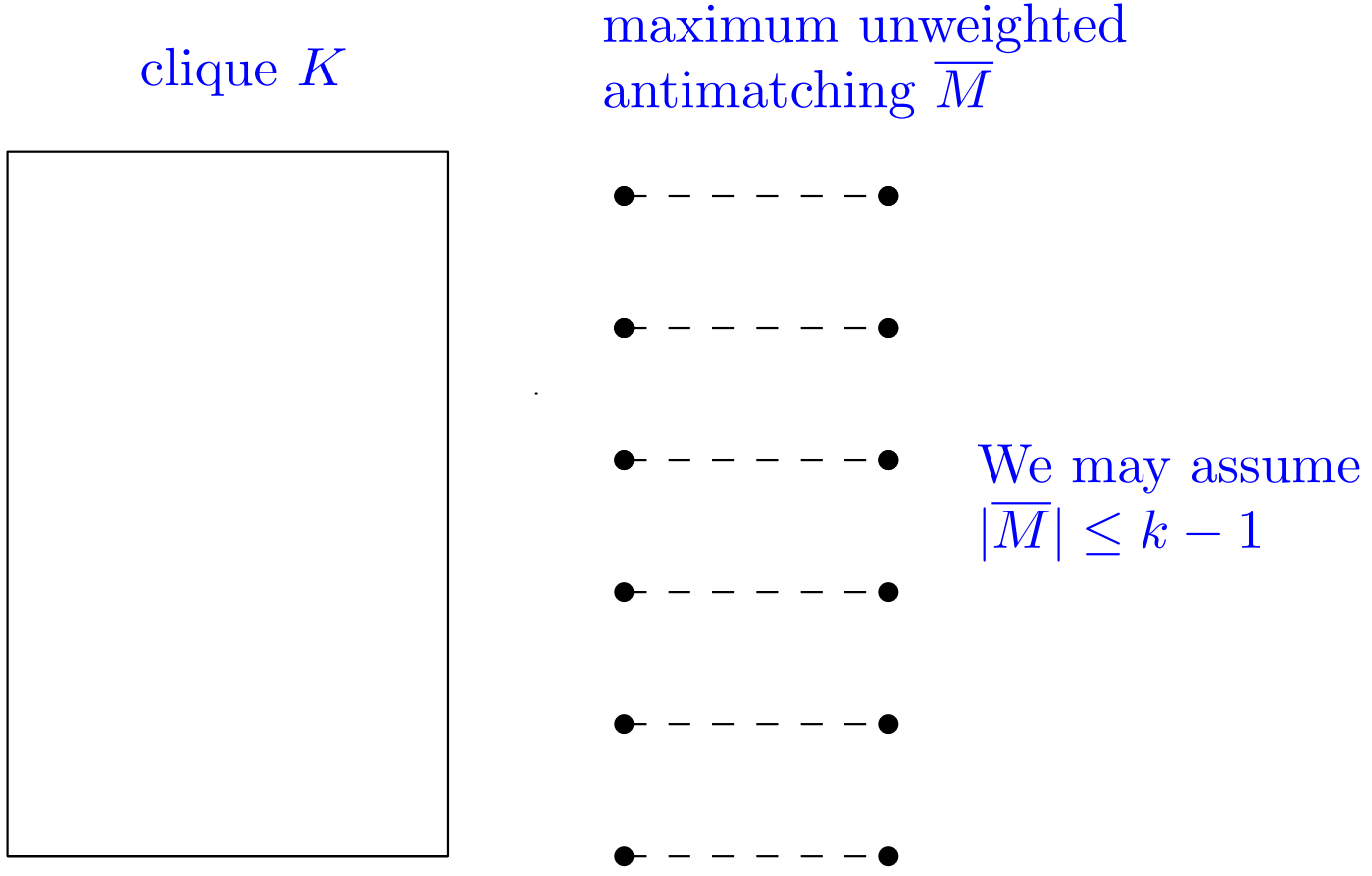
maximum unweighted antimatching $\overline{M}$



We may assume $|\overline{M}| \leq k-1$

If $|\overline{M}| \geq k$, we are dealing with a Yes-instance.

An exponential kernel for Dual Weighted Coloring

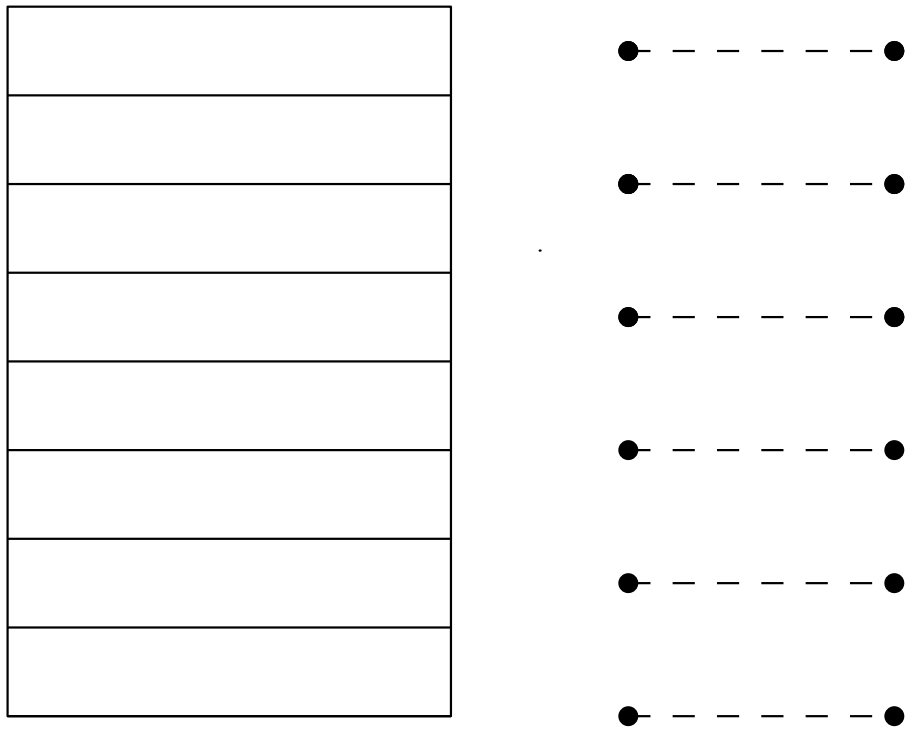


The complement of the antimatching $\overline{M}$ induces a clique K.

An exponential kernel for Dual Weighted Coloring

clique K

maximum unweighted antimatching $\overline{M}$

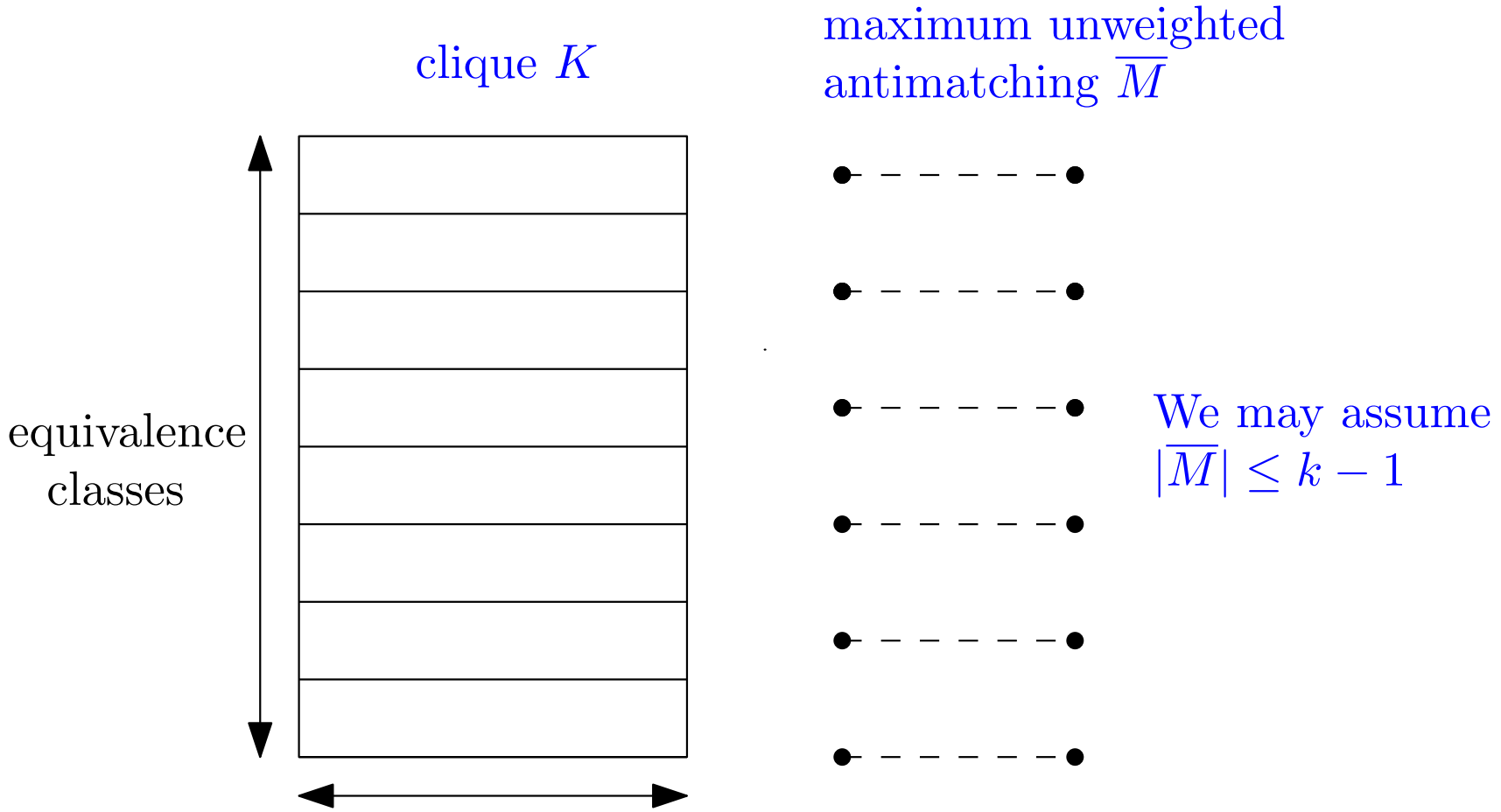


equivalence classes

We may assume $|\overline{M}| \leq k-1$

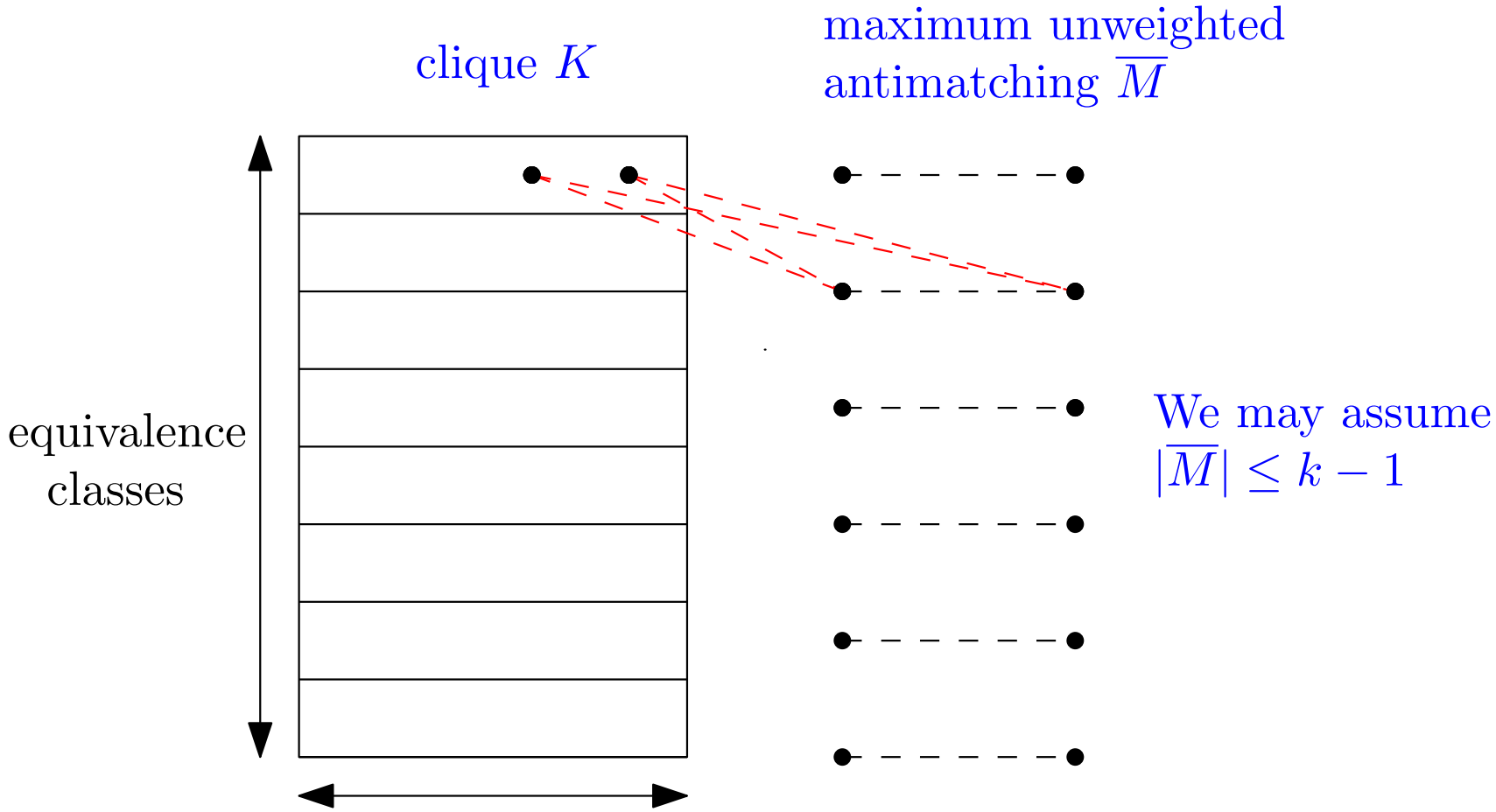
We partition K into equivalence classes w.r.t. the neighborhood in $\overline{M}$.

An exponential kernel for Dual Weighted Coloring



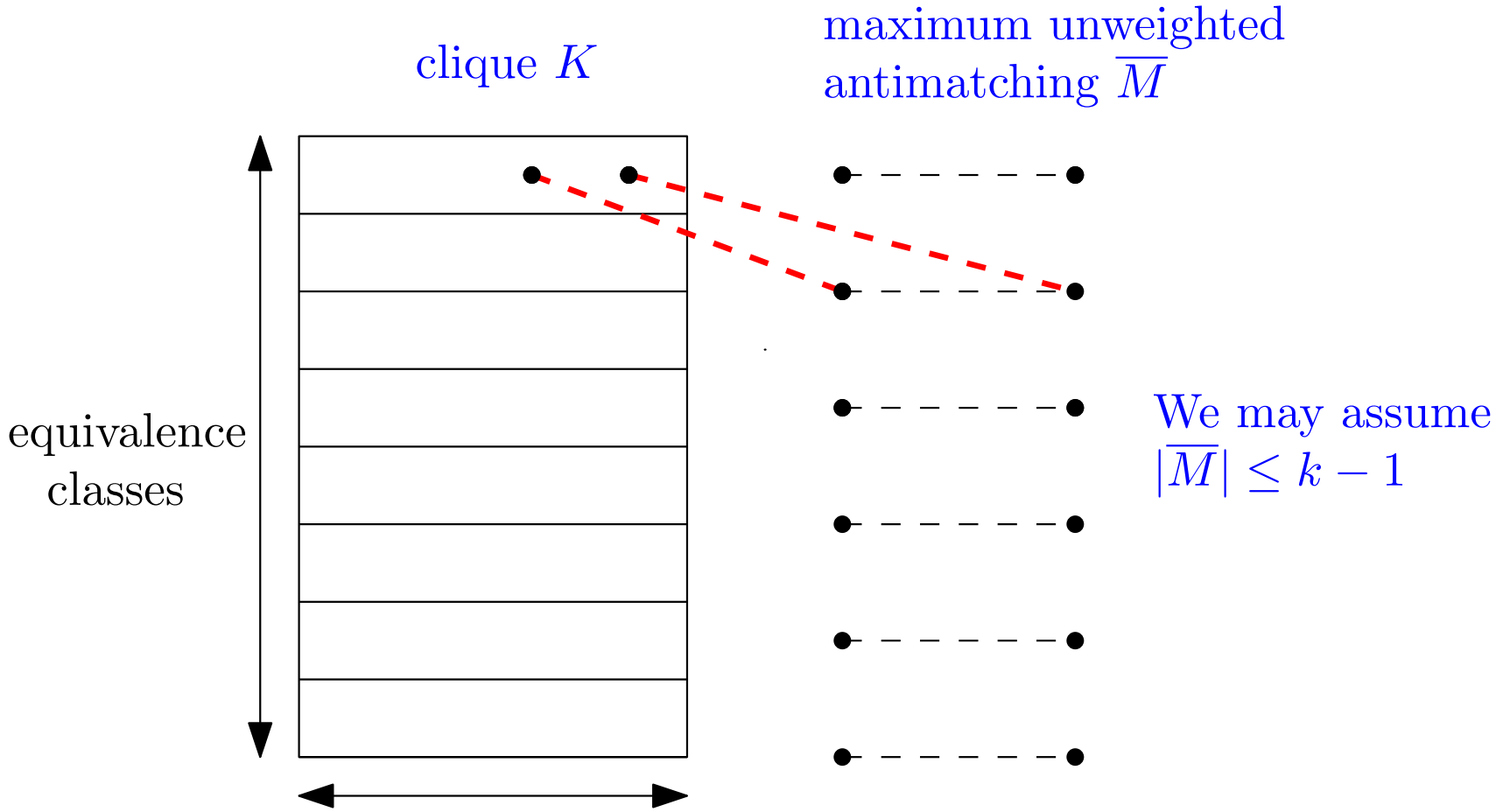
We have to bound the number and the size of the equivalence classes.

An exponential kernel for Dual Weighted Coloring



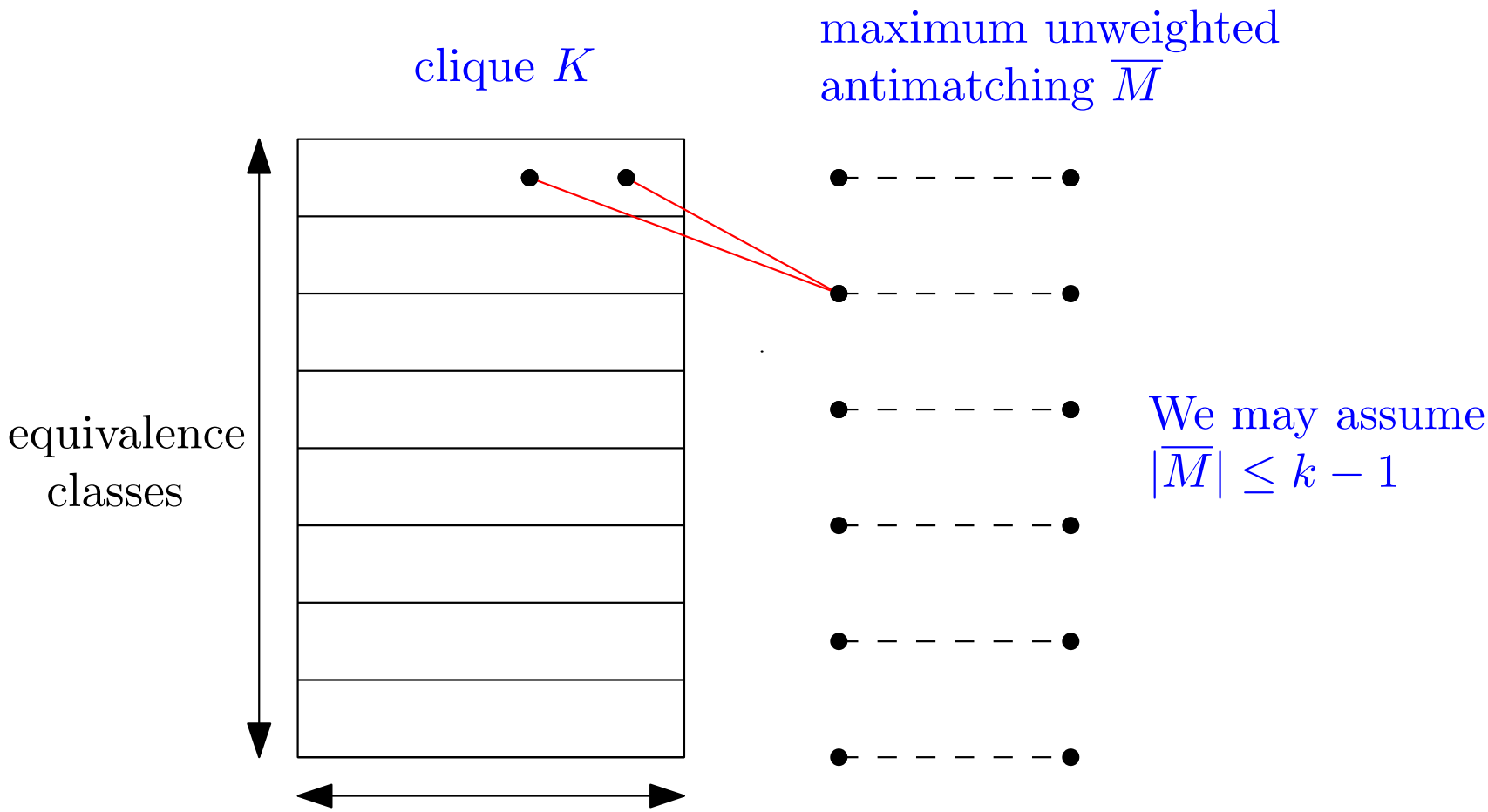
An equivalence class cannot miss both endvertices of a non-edge.

An exponential kernel for Dual Weighted Coloring



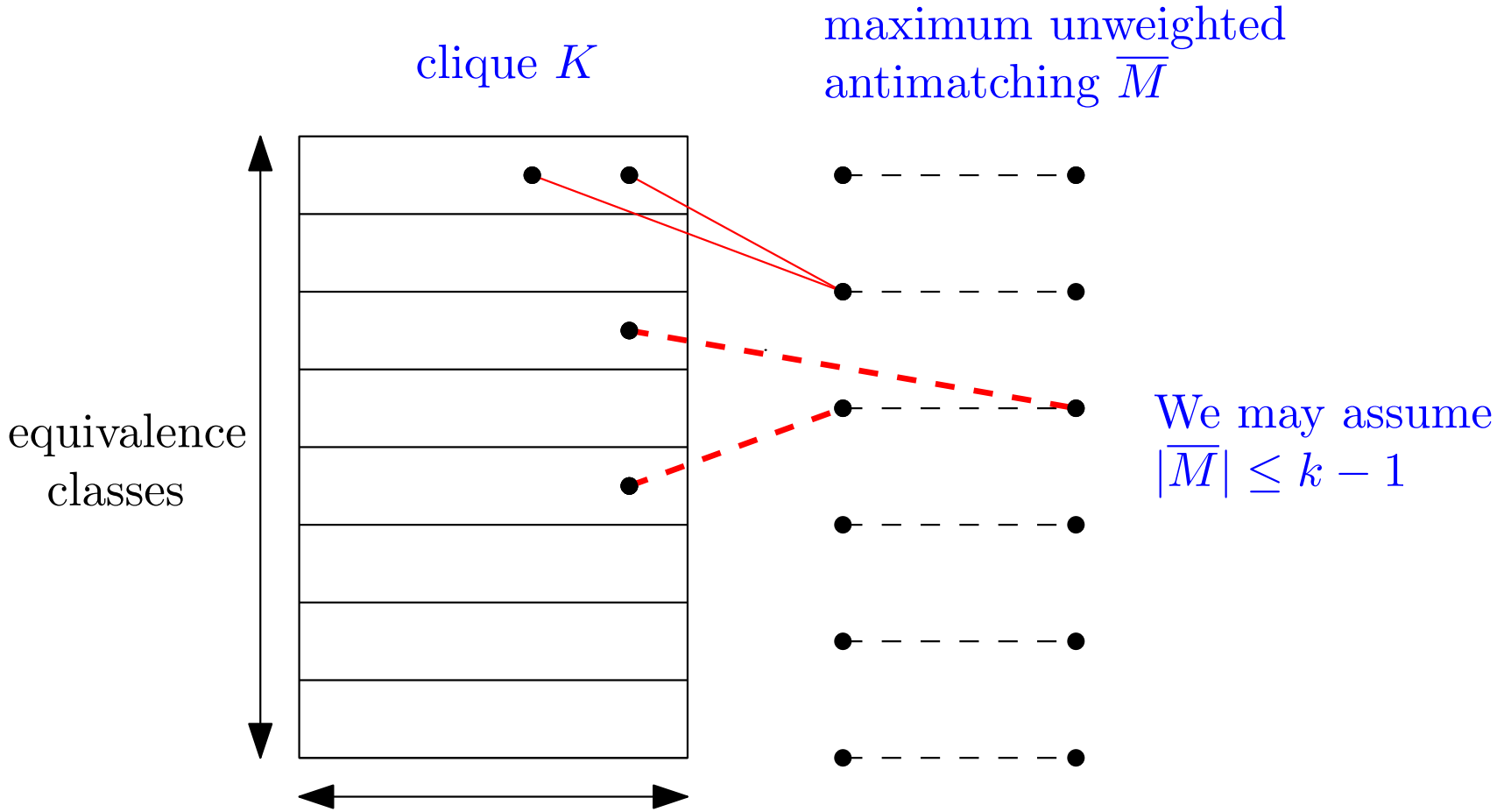
An equivalence class cannot miss both endvertices of a non-edge.

An exponential kernel for Dual Weighted Coloring



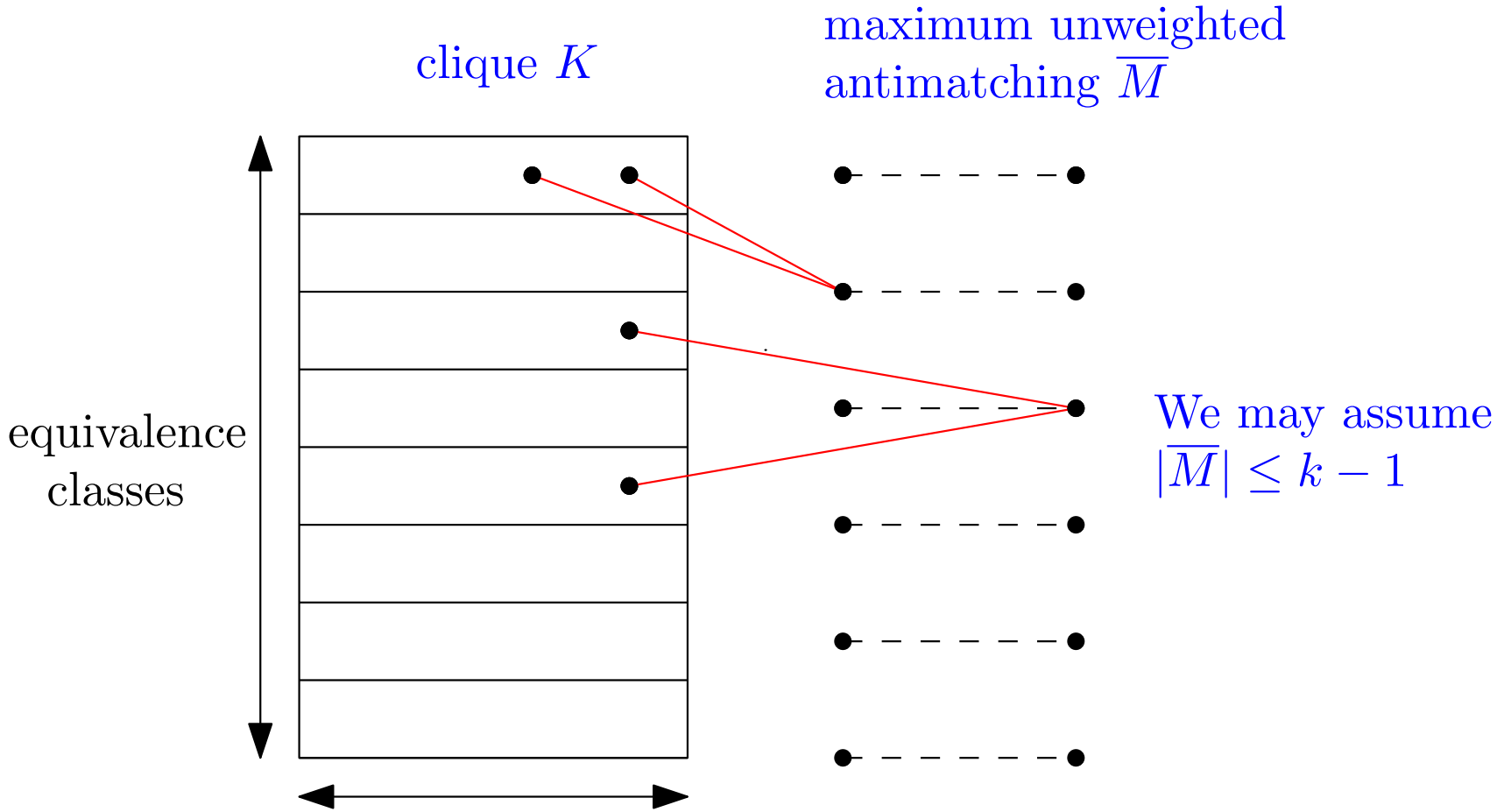
An equivalence class cannot miss both endvertices of a non-edge.

An exponential kernel for Dual Weighted Coloring



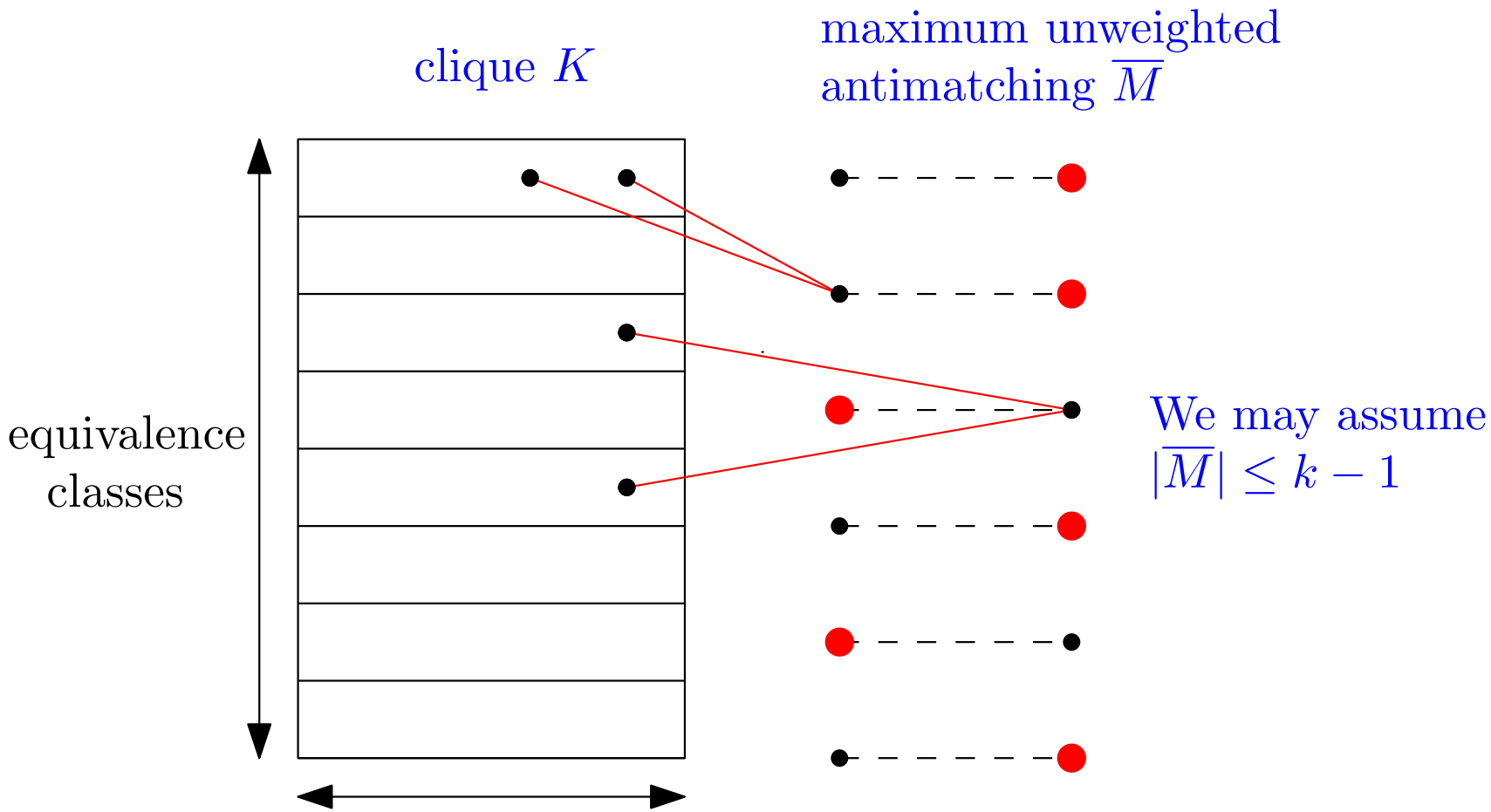
Two equivalence classes cannot miss different endvertices of a non-edge.

An exponential kernel for Dual Weighted Coloring



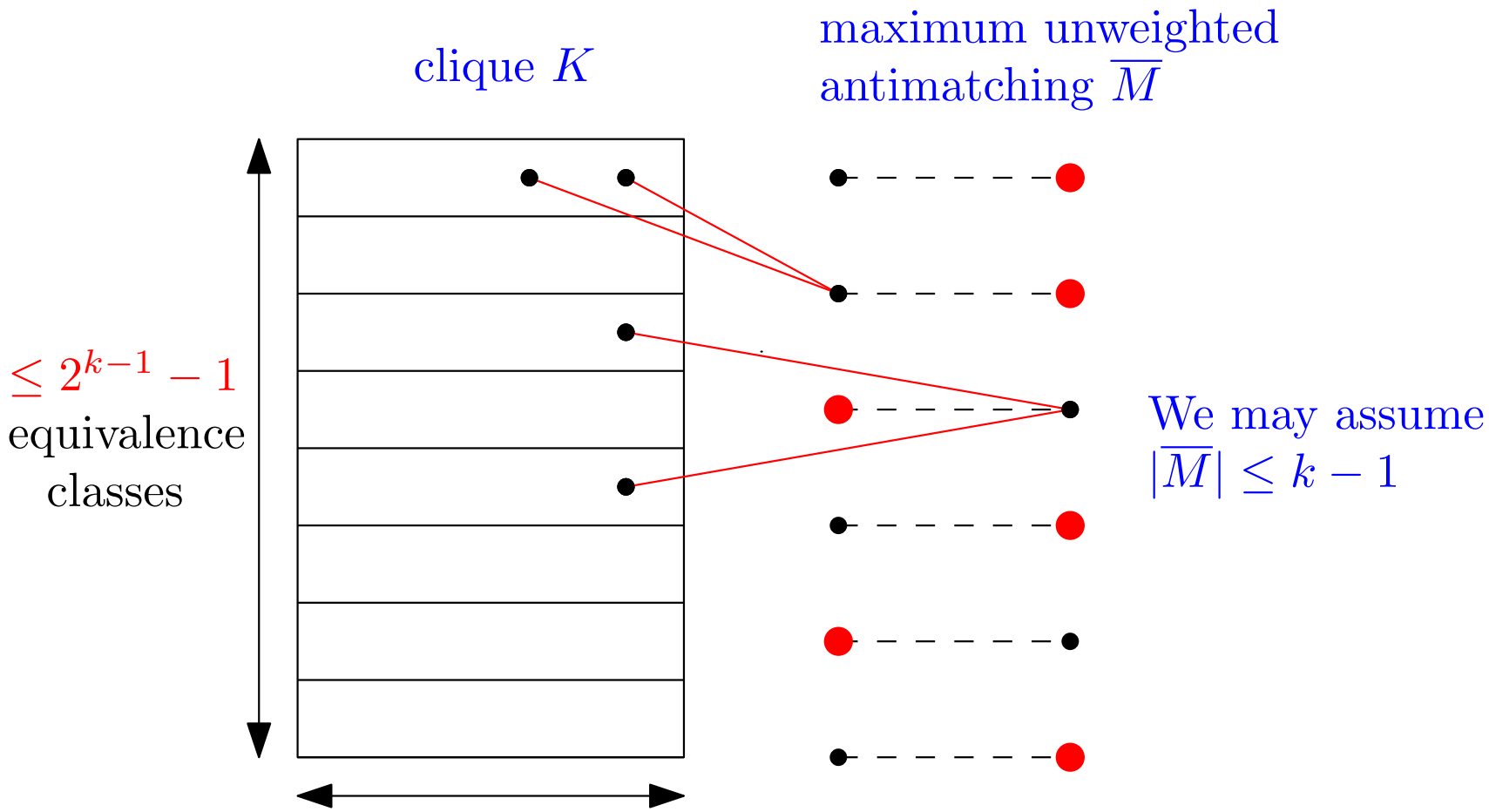
Two equivalence classes cannot miss different endvertices of a non-edge.

An exponential kernel for Dual Weighted Coloring



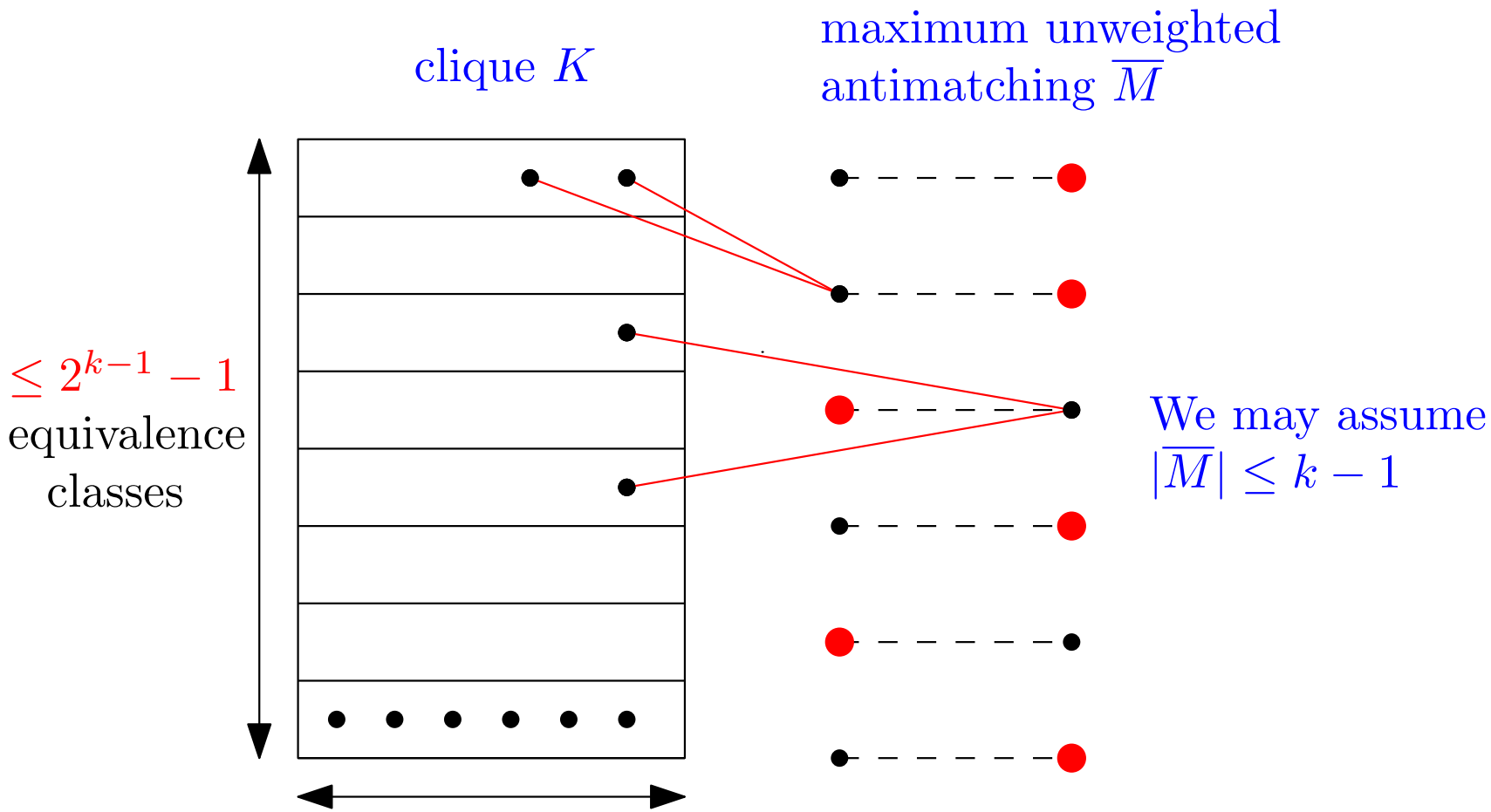
Every non-edge has a unique "potential non-neighbor" for the classes.

An exponential kernel for Dual Weighted Coloring



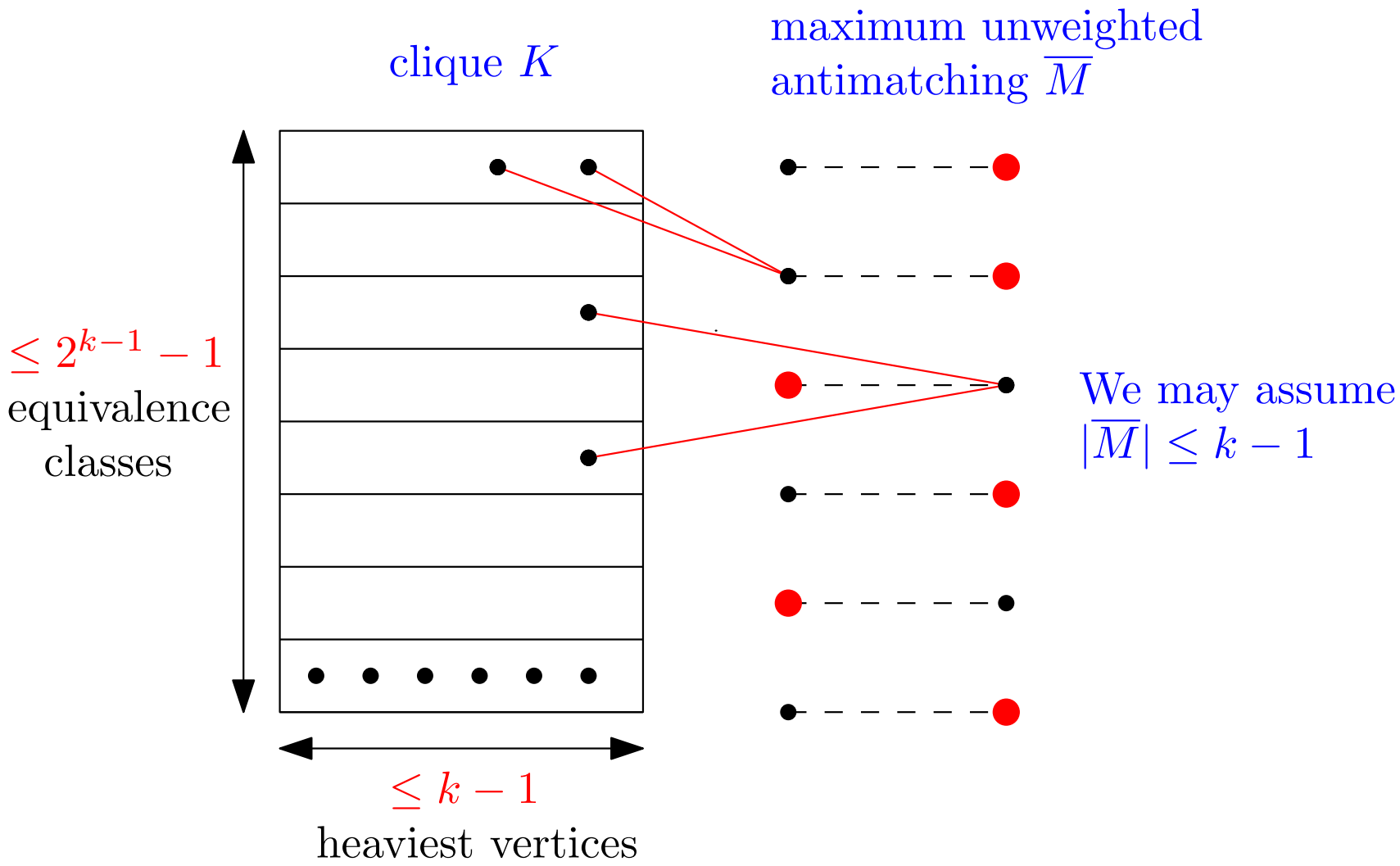
Hence, the number of equivalence classes is at most $2^{k-1} - 1$.

An exponential kernel for Dual Weighted Coloring



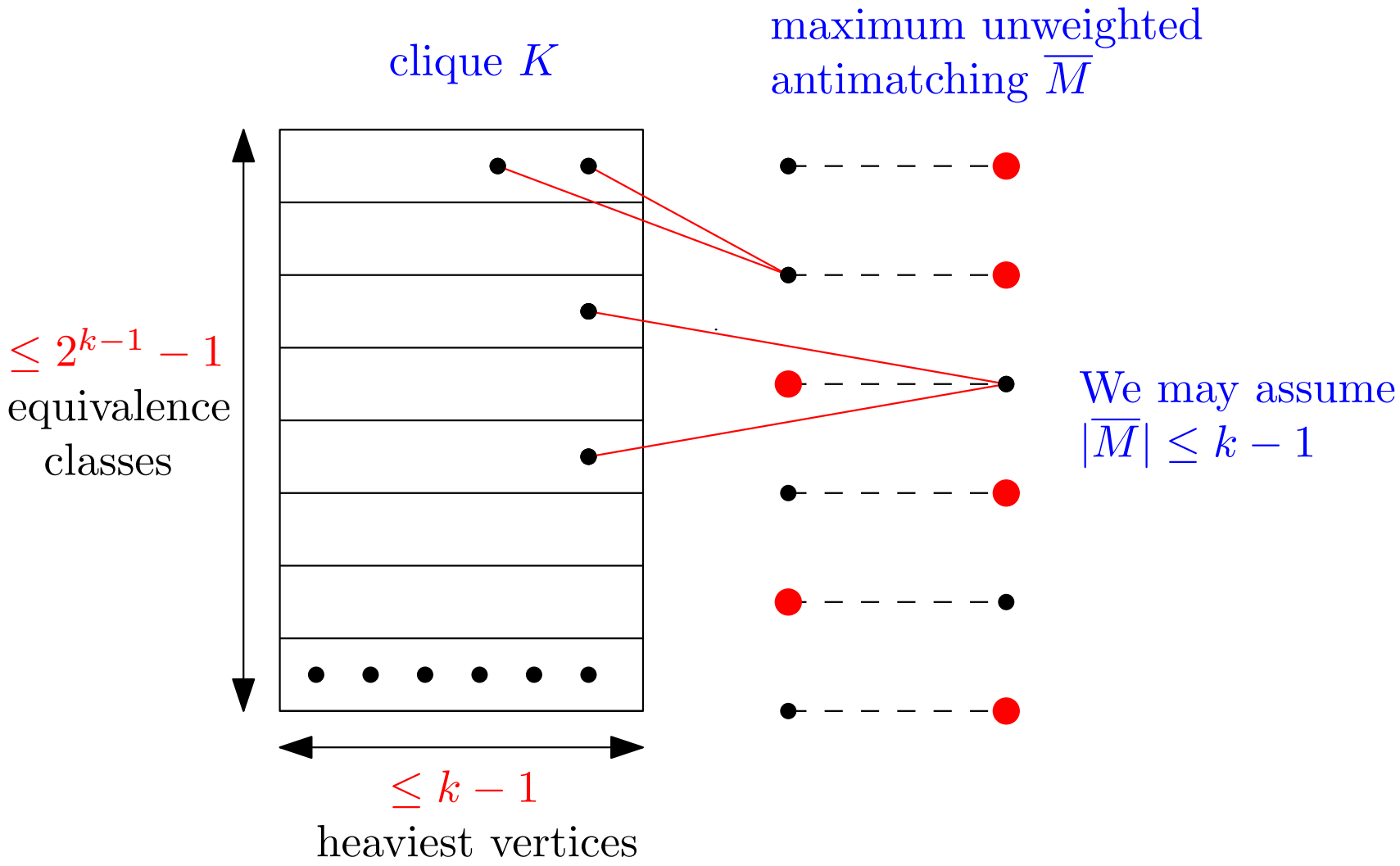
It remains to bound the size of each equivalence class.

An exponential kernel for Dual Weighted Coloring



It is safe to keep only the $k - 1$ heaviest vertices of each equivalence class.

An exponential kernel for Dual Weighted Coloring



Thus, $|V(G)| \leq |V(\overline{M})| + |K| \ \leq \ 2(k-1) + (2^{k-1} - 1)(k-1)$.

Ruling out polynomial kernels

Theorem

Dual Weighted Coloring *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly, *even on split graphs with only two different weights.*

Ruling out polynomial kernels

Theorem

Dual Weighted Coloring *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly, *even on split graphs with only two different weights.*

We present a polynomial parameter transformation from Set Cover parameterized by the size of the universe, known not to admit polynomial kernels unless NP $\subseteq$ coNP/poly. [Dom, Lokshtanov, Saurabh. 2014]

Ruling out polynomial kernels

Theorem

Dual Weighted Coloring *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly, *even on split graphs with only two different weights.*

We present a polynomial parameter transformation from Set Cover parameterized by the size of the universe, known not to admit polynomial kernels unless NP $\subseteq$ coNP/poly. [Dom, Lokshtanov, Saurabh. 2014]

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

A polynomial parameter transformation from A to B is an algorithm s.t.:

1. (x, k) is a Yes-instance of $A \Leftrightarrow (x', k')$ is a Yes-instance of B.
2. $k' \leq \text{poly}(k)$.

[Bodlaender, Thomassé, Yeo. 2011]

Ruling out polynomial kernels

Theorem

Dual Weighted Coloring *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly, *even on split graphs with only two different weights.*

We present a polynomial parameter transformation from Set Cover parameterized by the size of the universe, known not to admit polynomial kernels unless NP $\subseteq$ coNP/poly. [Dom, Lokshtanov, Saurabh. 2014]

Our reduction is an appropriate modification of a reduction to prove the NP-hardness of Weighted Coloring on split graphs:

only the weights change.

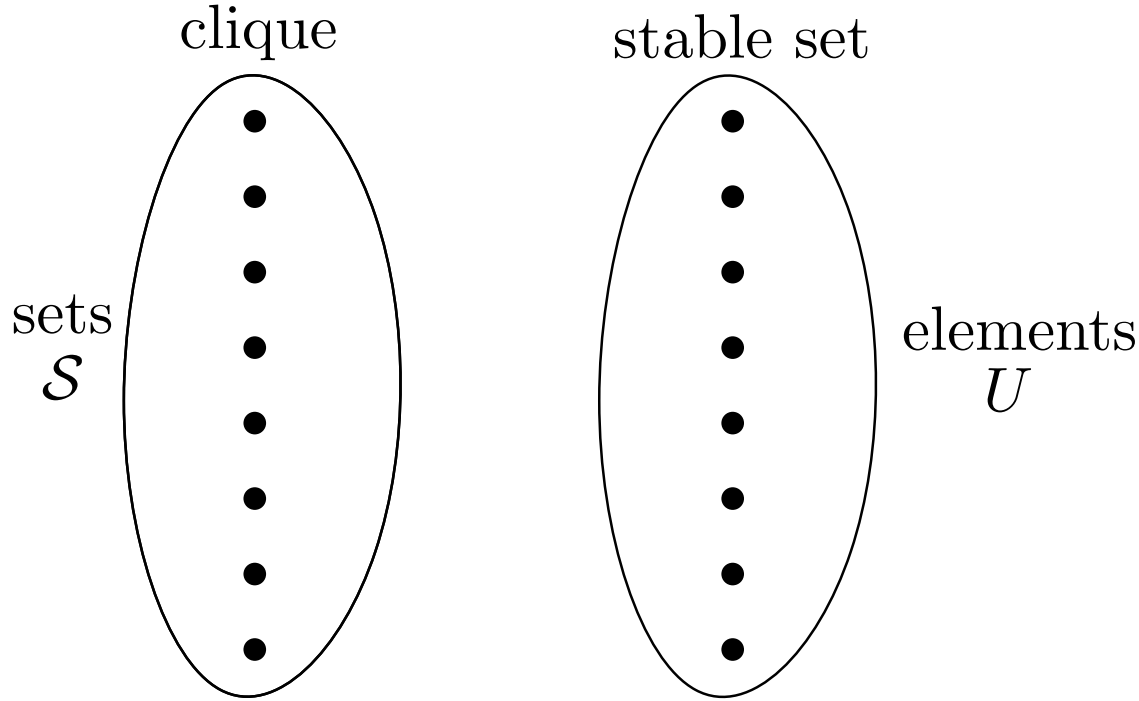
[Demange, de Werra, Monnot, Paschos. 2002]

Polynomial parameter transformation from Set Cover

We are given an instance $(U, \mathcal{S}, k, \ell)$ of Set Cover,
where $\mathcal{S}$ is a family of sets over a universe U of size k (the parameter).
Question: $\exists$? $\mathcal{S}' \subseteq \mathcal{S}$ of at most ℓ sets covering all the elements of U.

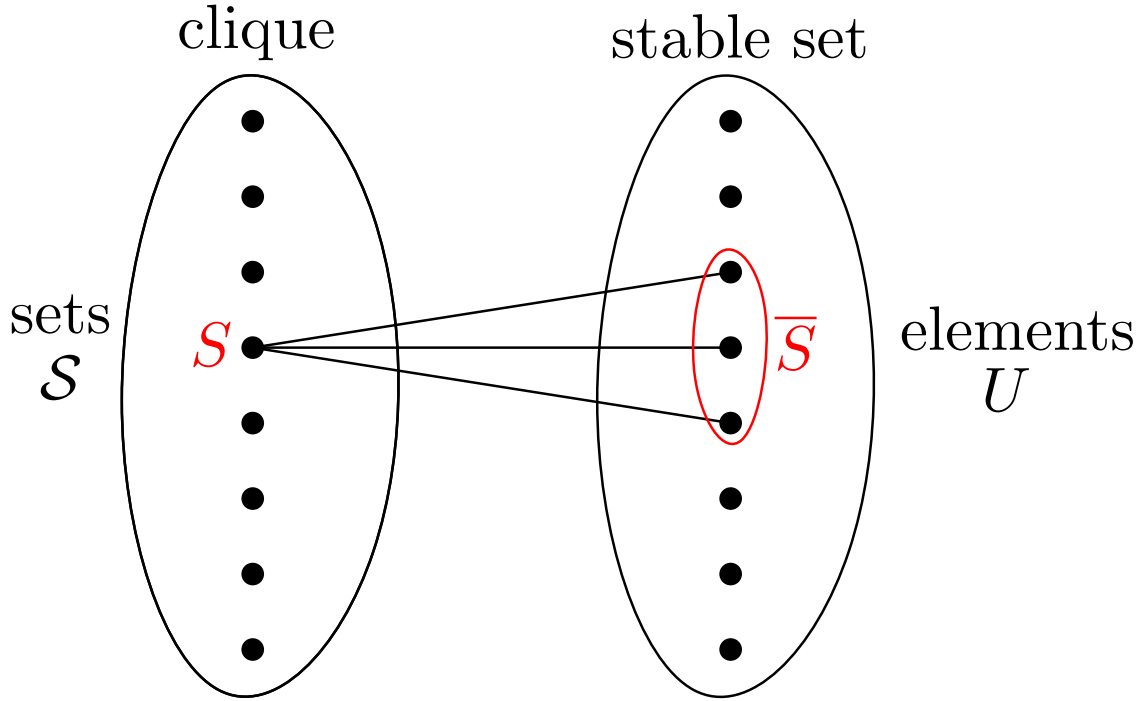
Polynomial parameter transformation from Set Cover

We are given an instance $(U, \mathcal{S}, k, \ell)$ of Set Cover,
where $\mathcal{S}$ is a family of sets over a universe U of size k (the parameter).
Question: $\exists$? $\mathcal{S}' \subseteq \mathcal{S}$ of at most ℓ sets covering all the elements of U.



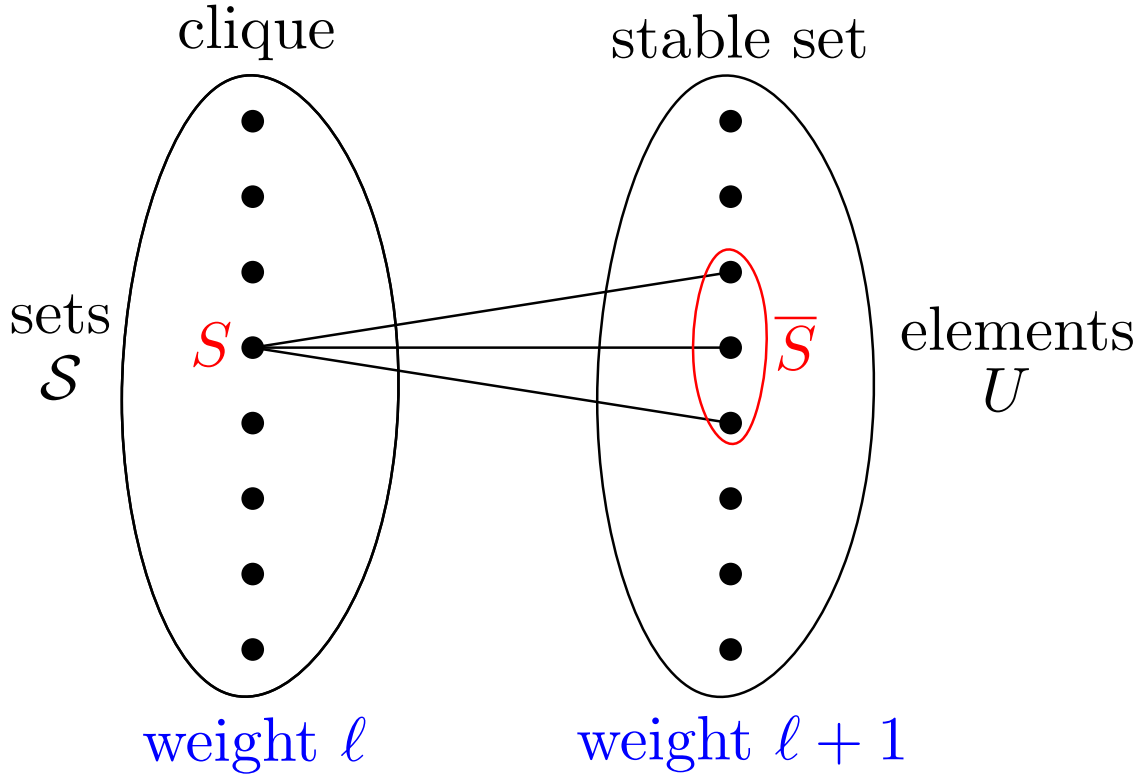
Polynomial parameter transformation from Set Cover

We are given an instance $(U, \mathcal{S}, k, \ell)$ of Set Cover,
where $\mathcal{S}$ is a family of sets over a universe U of size k (the parameter).
Question: $\exists$? $\mathcal{S}' \subseteq \mathcal{S}$ of at most ℓ sets covering all the elements of U.



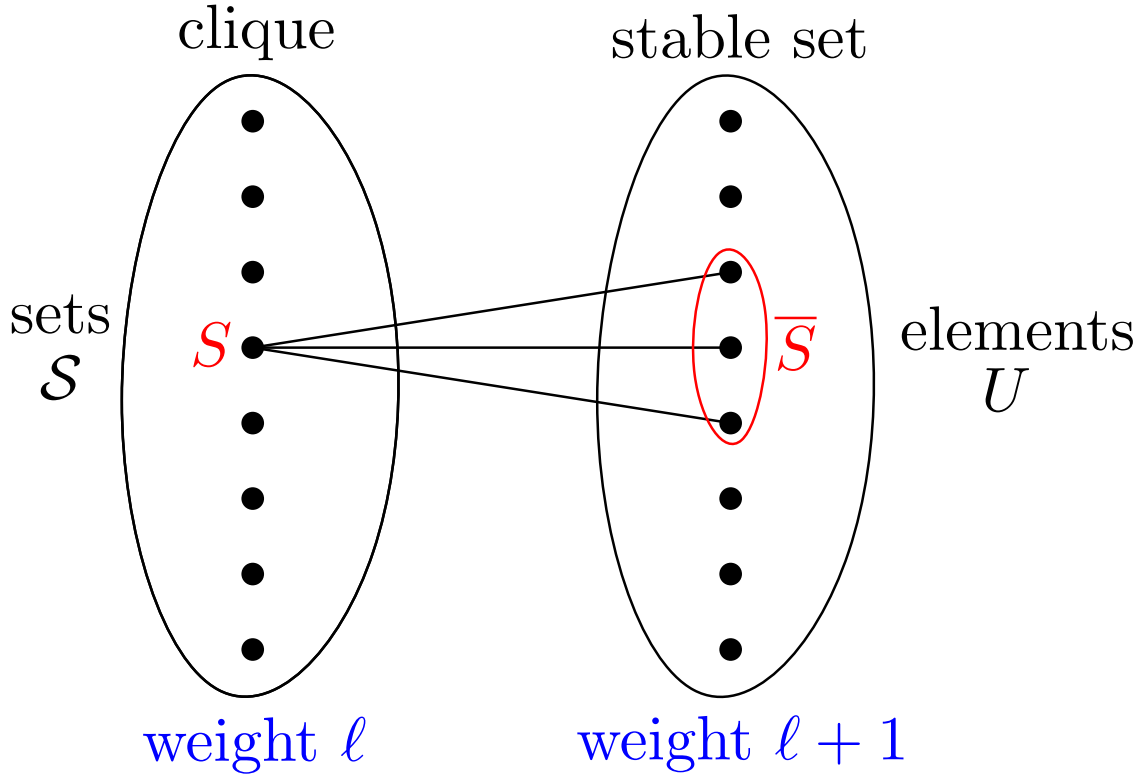
Polynomial parameter transformation from Set Cover

We are given an instance $(U, \mathcal{S}, k, \ell)$ of Set Cover,
where $\mathcal{S}$ is a family of sets over a universe U of size k (the parameter).
Question: $\exists$? $\mathcal{S}' \subseteq \mathcal{S}$ of at most ℓ sets covering all the elements of U.



Polynomial parameter transformation from Set Cover

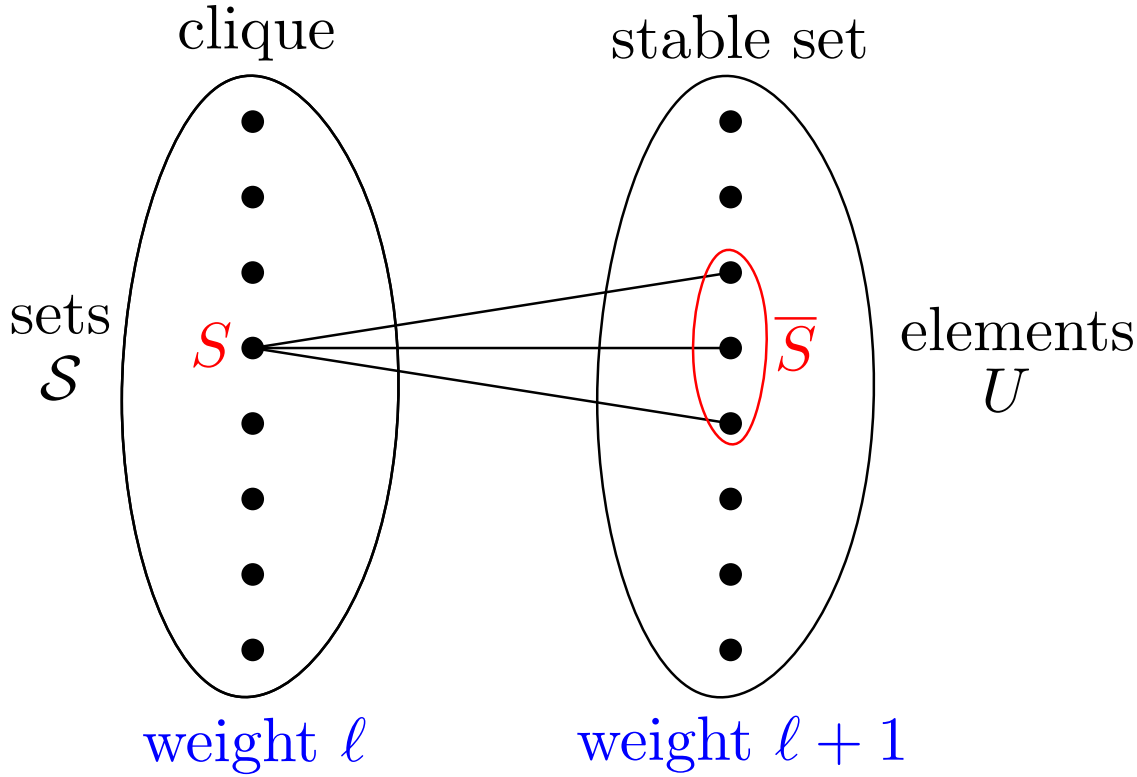
We are given an instance $(U, \mathcal{S}, k, \ell)$ of Set Cover,
where $\mathcal{S}$ is a family of sets over a universe U of size k (the parameter).
Question: $\exists$? $\mathcal{S}' \subseteq \mathcal{S}$ of at most ℓ sets covering all the elements of U.



We build and instance (G, w, k'), with $k' = k(\ell + 1) - \ell = \mathcal{O}(k^2)$.

Polynomial parameter transformation from Set Cover

We are given an instance $(U, \mathcal{S}, k, \ell)$ of Set Cover,
where $\mathcal{S}$ is a family of sets over a universe U of size k (the parameter).
Question: $\exists$? $\mathcal{S}' \subseteq \mathcal{S}$ of at most ℓ sets covering all the elements of U.



$(U, \mathcal{S}, k, \ell)$ satisfiable $\Leftrightarrow$ $\sigma(G, w) \leq \sum_{v \in V(G)} w(v) - k' = |\mathcal{S}| \cdot \ell + \ell$.

Exploiting the relation with Set Cover

Dual Weighted Coloring seems to be strongly related to Set Cover...

Exploiting the relation with Set Cover

Dual Weighted Coloring seems to be strongly related to Set Cover... and also to d-Set Cover:

Proposition

Dual Weighted Coloring *restricted to split graphs such that each vertex in the clique has at most d non-neighbors in the stable set, for some constant $d \geq 2$, admits a kernel with $\mathcal{O}(k^d)$ vertices.*

Exploiting the relation with Set Cover

Dual Weighted Coloring seems to be strongly related to Set Cover... and also to d-Set Cover:

Proposition

Dual Weighted Coloring *restricted to split graphs such that each vertex in the clique has at most d non-neighbors in the stable set, for some constant $d \geq 2$, admits a kernel with $\mathcal{O}(k^d)$ vertices.*

Furthermore, for any $\varepsilon > 0$, a kernel with $\mathcal{O}(k^{\frac{d-3}{2}-\varepsilon})$ vertices does not exist unless NP $\subseteq$ coNP/poly.

Using the kernel lower bound for d-Set Cover. [Hermelin, Wu. 2012]

Some questions for further research

- What is the complexity of Weighted Coloring for general P_4-sparse graphs?

Some questions for further research

- What is the complexity of Weighted Coloring for general P_4-sparse graphs?
- We presented an FPT algorithm for Dual Weighted Coloring running in time $9^k \cdot n^{\mathcal{O}(1)}$.

Some questions for further research

- What is the complexity of Weighted Coloring for general P_4-sparse graphs?

- We presented an FPT algorithm for Dual Weighted Coloring running in time $9^k \cdot n^{\mathcal{O}(1)}$.

  Improve the running time and/or prove lower bounds under the SETH.

Some questions for further research

- What is the complexity of Weighted Coloring for general P_4-sparse graphs?

- We presented an FPT algorithm for Dual Weighted Coloring running in time $9^k \cdot n^{\mathcal{O}(1)}$.

  Improve the running time and/or prove lower bounds under the SETH.

- Can the cubic kernel on interval graphs be improved?

Some questions for further research

- What is the complexity of Weighted Coloring for general P_4-sparse graphs?
- We presented an FPT algorithm for Dual Weighted Coloring running in time $9^k \cdot n^{\mathcal{O}(1)}$.

  Improve the running time and/or prove lower bounds under the SETH.
- Can the cubic kernel on interval graphs be improved?
- Identify other classes of (dense) graphs allowing for polynomial kernels.

Thank you!