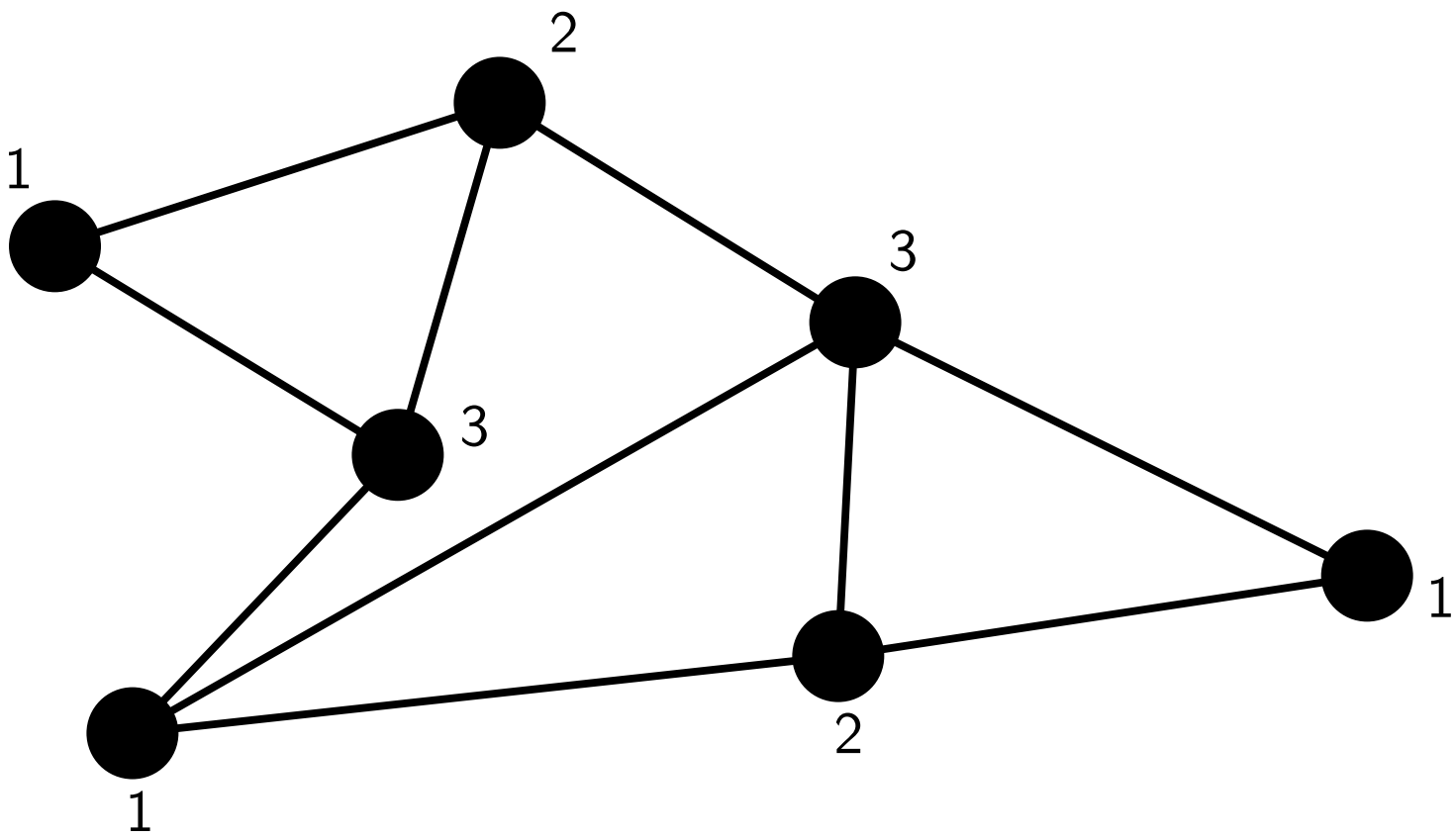
Tight lower bound for the Channel Assignment problem

Arkadiusz Socała

University of Warsaw

Graph Coloring

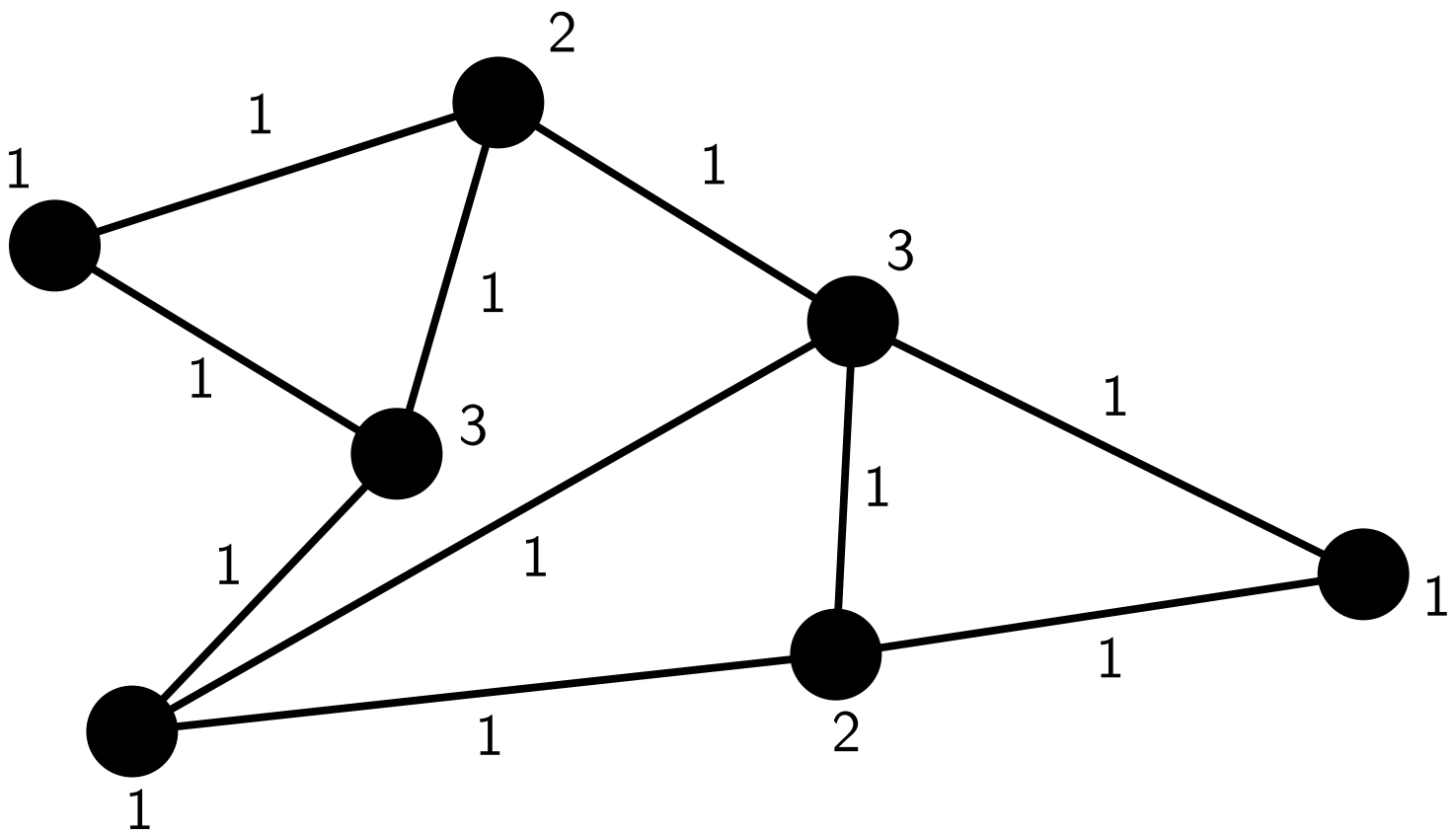
Recall the Graph Coloring problem.



- coloring $c : V \to \{1, \ldots, s\}$
- $c(u) \neq c(v)$ for every edge $uv \in E$

Graph Coloring

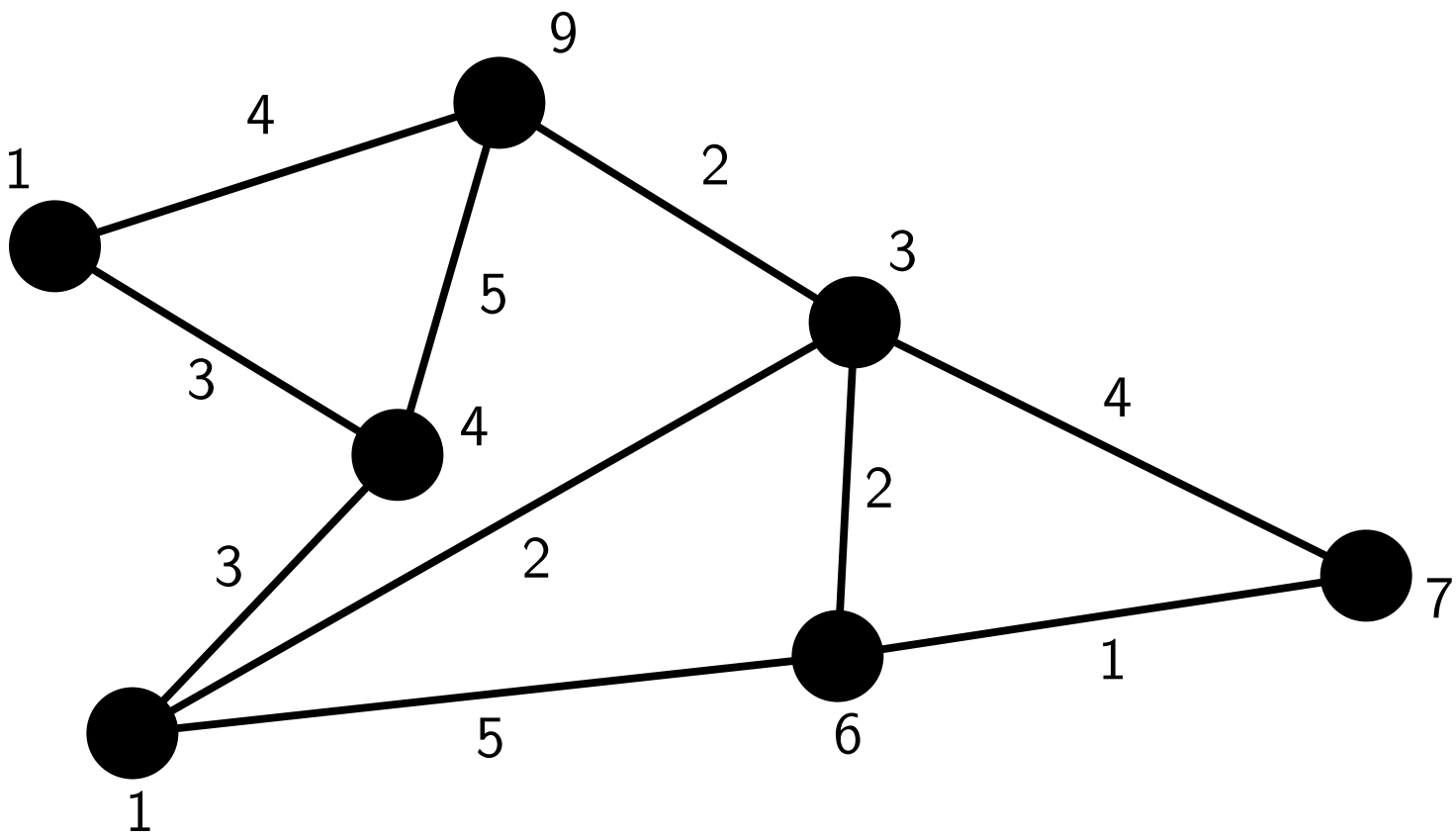
Recall the Graph Coloring problem.



- coloring $c : V \to \{1, \ldots, s\}$
- $|c(u) - c(v)| \geq 1$ for every edge $uv \in E$

Channel Assignment

Weights on edges mean minimum allowed color differences.



- coloring $c : V \to \{1, \ldots, s\}$
- $|c(u) - c(v)| \geq w(uv)$ for every edge $uv \in E$

The Channel Assignment problem.

Input

- graph $G(V, E)$
- weight function $w : E \to \mathbb{N}_+$.

Definitions

- An assignment $c : V \to \{1, \ldots, s\}$ is called *proper* when $\forall_{uv \in E} |c(u) - c(v)| \geq w(uv)$.
- The number s is called the *span* of c.

Problem

Find a proper assignment of minimum span.

The Channel Assignment problem.

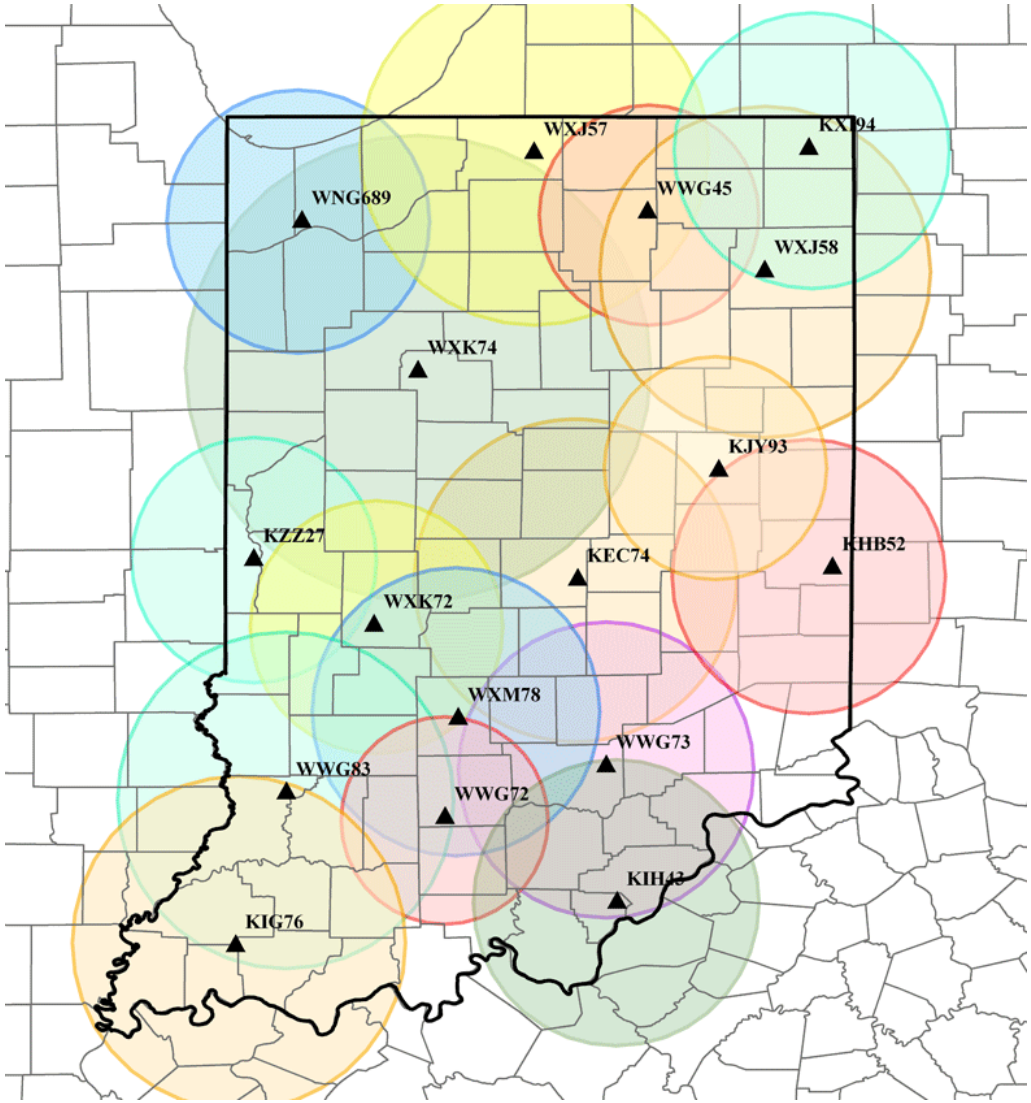
Motivation

Assignment of the radio frequencies:

- n radio emitters
- they interfere each other
- minimize the range of the used frequencies (span)

Introduced by

- Hale in 1980.



Algorithms for Channel Assignment

General version

- $O^*(n!)$-time (McDiarmid, 2003)
- **Our result:** There is no $2^{o(n \log n)}$-time algorithm (under ETH)

ℓ-bounded version

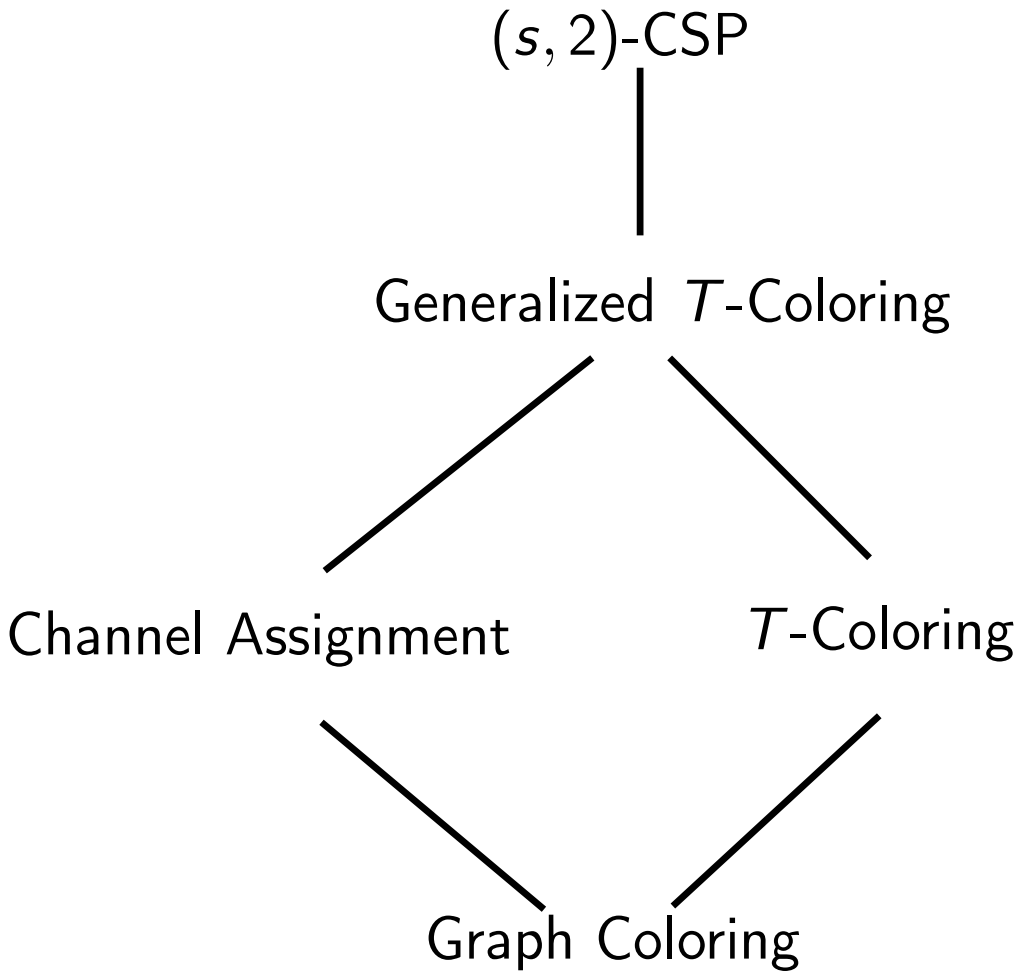
Assume $\forall_{uv \in E}\ w(uv) \leq \ell$.

- $O^*((2\ell+1)^n)$-time (McDiarmid, 2003)
- $O^*((\ell+2)^n)$-time (Kral, 2005)
- $O^*((\ell+1)^n)$-time (Cygan and Kowalik, 2011)
- $O^*((2\sqrt{\ell+1})^n)$-time (Kowalik and S, 2014)

These are all dynamic programming.

All algorithms above can be modified to **count** proper colorings.

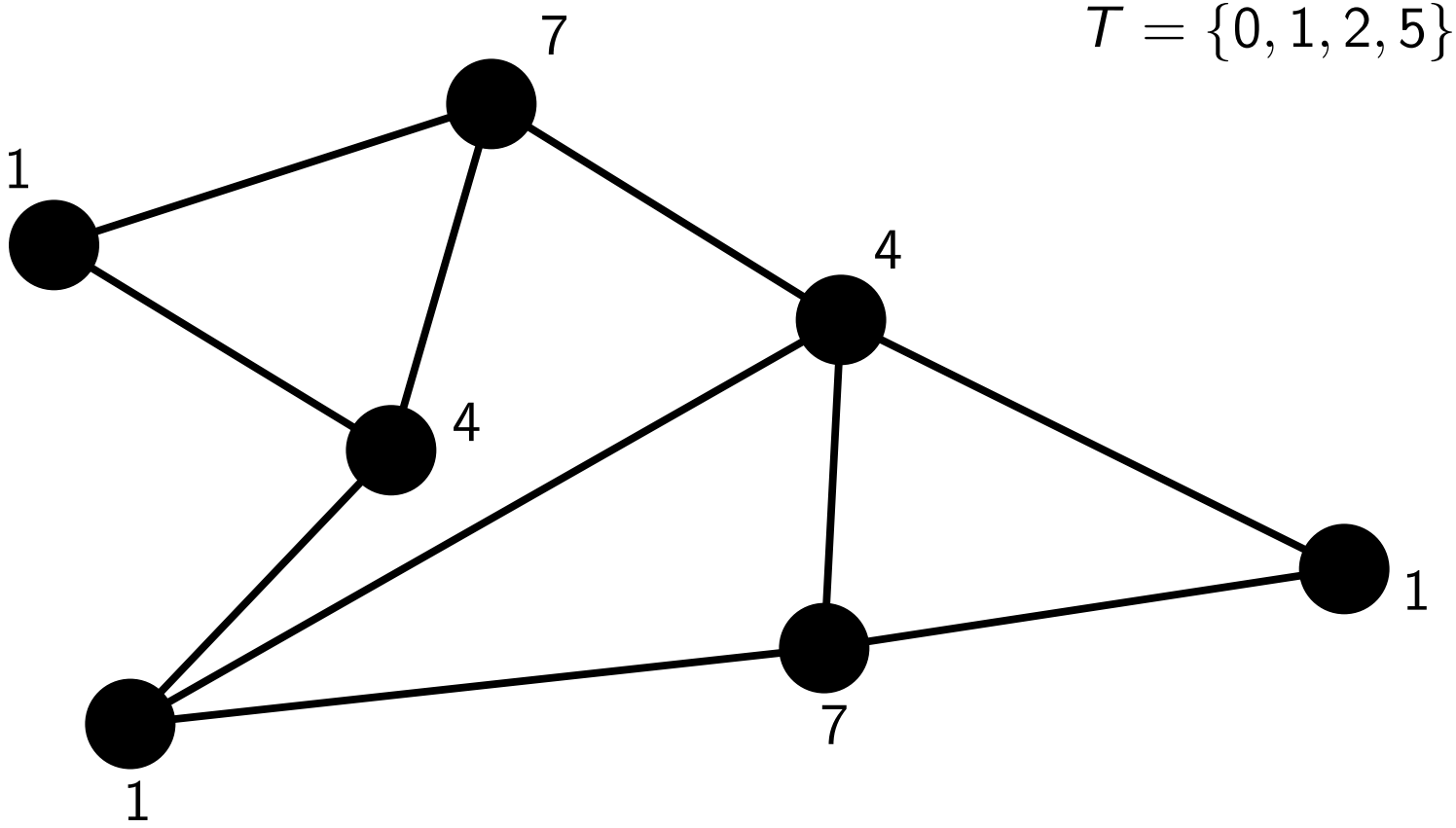
CSP Hierarchy



T-Coloring

List of forbidden differences.

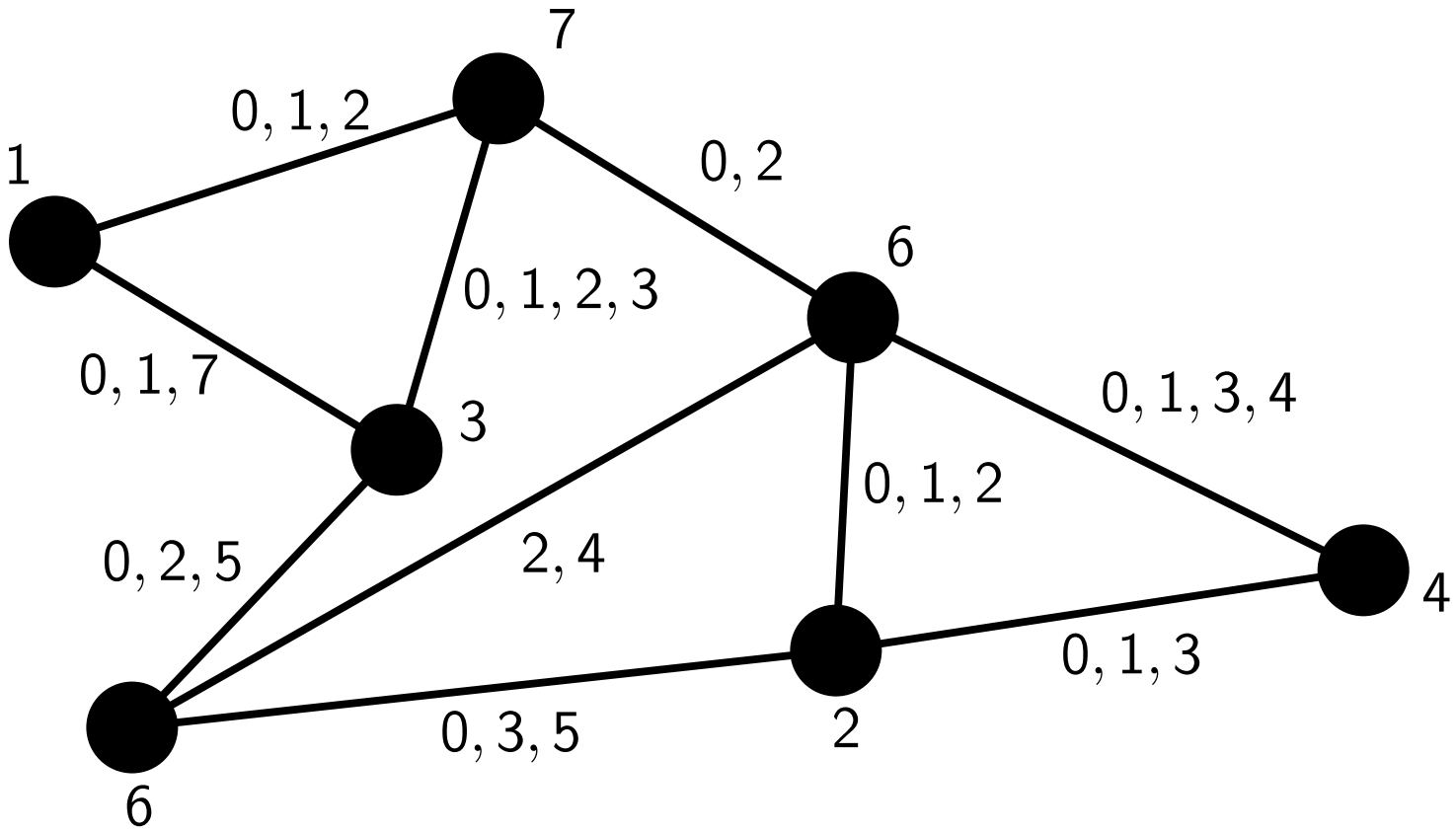
$T = \{0, 1, 2, 5\}$



- coloring $c : V \to \{1, \ldots, s\}$
- $|c(u) - c(v)| \notin T$ for every edge $uv \in E$

Generalized T-Coloring

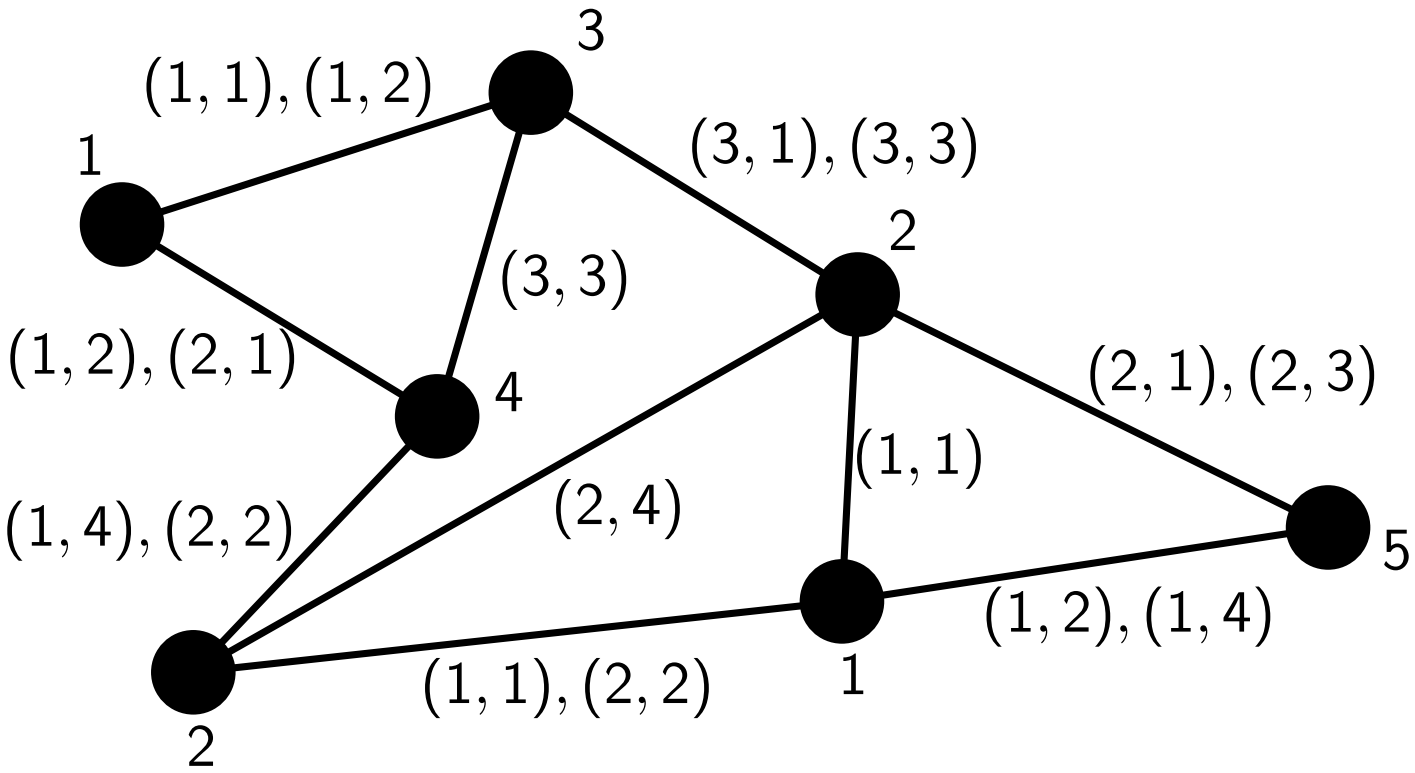
Lists of forbidden differences.



- coloring $c : V \to \{1, \ldots, s\}$
- $|c(u) - c(v)| \notin T(uv)$ for every edge $uv \in E$

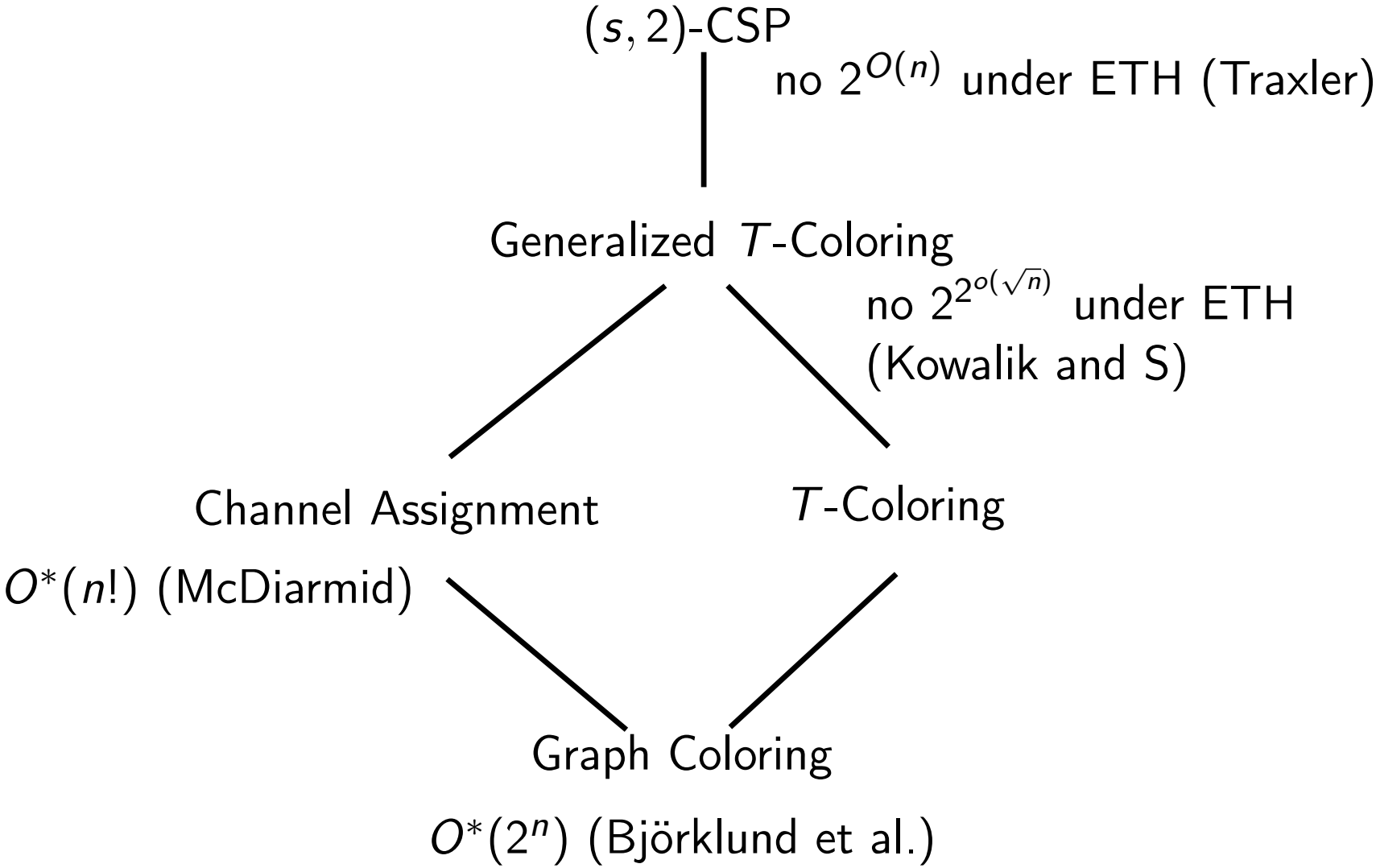
$(s, 2)$-CSP (Constraint Satisfaction Problem)

Lists of forbidden pairs of colors.

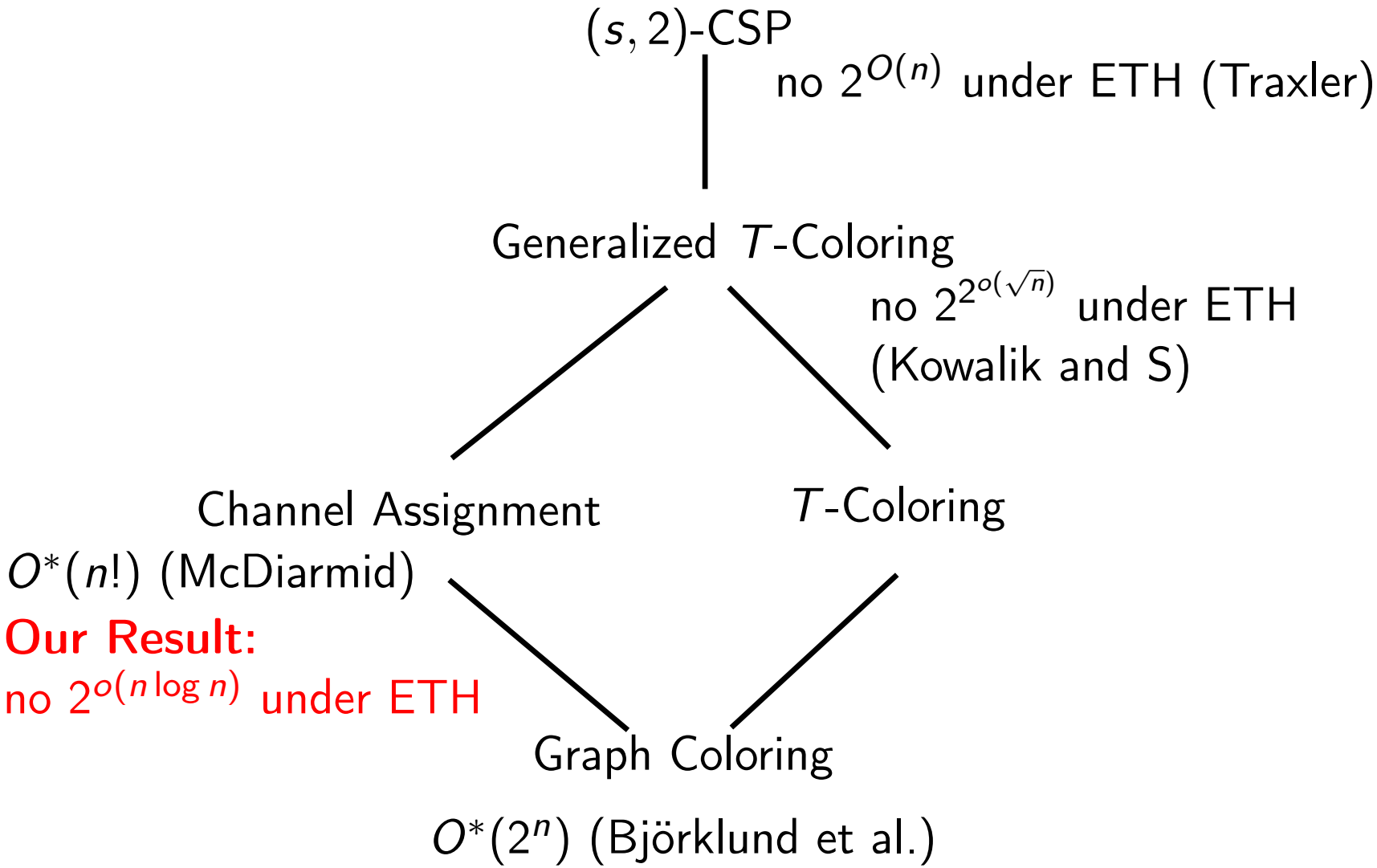


- coloring $c : V \to \{1, \ldots, s\}$
- $(c(u), c(v)) \notin C(uv)$ for every edge $uv \in E$

CSP Hierarchy



CSP Hierarchy



Summary of our results

- Channel Assignment cannot be solved in $2^{o(n \log n)}$ under ETH (tight!)
- nor in $2^{n \cdot o(\log \log \ell)}$

From 3-CNF-SAT to Family Intersection

From 3-CNF-SAT to Family Intersection

Family Intersection:

- Input: We are given two matrices.
- We have to pick exactly one element from every row.
- Question: Is it possible to pick elements in such a way that both sums are equal?

Choice 1:	1	14	1
Choice 2:	7	6	10
Choice 3:	6	4	7
Choice 4:	2	0	14

Choice 1:	5	7	5	4
Choice 2:	2	8	7	7
Choice 3:	3	8	8	9

From 3-CNF-SAT to Family Intersection

Family Intersection:

- Input: We are given two matrices.
- We have to pick exactly one element from every row.
- Question: Is it possible to pick elements in such a way that both sums are equal?

Choice 1:	1	14	1
Choice 2:	7	6	10
Choice 3:	6	4	7
Choice 4:	2	0	14

Choice 1:	5	7	5	4
Choice 2:	2	8	7	7
Choice 3:	3	8	8	9

e.g. $1 + 7 + 7 + 0 = 5 + 7 + 3$

From 3-CNF-SAT to Family Intersection

An example for a 2-CNF-SAT formula $\varphi = (\alpha \vee \beta) \wedge (\neg\alpha \vee \gamma)$:

- We number all the occurrences $(\alpha_1 \vee \beta_2) \wedge (\neg\alpha_3 \vee \gamma_4)$
- We interpret 4-bit numbers as the valuations of the occurrences

	False	True
α:	0000_2	1010_2
β:	0000_2	0100_2
γ:	0000_2	0001_2

	1st	2nd	1st and 2nd
$\alpha_1 \vee \beta_2$:	1000_2	0100_2	1100_2
$\neg\alpha_3 \vee \gamma_4$:	0000_2	0011_2	0001_2

From 3-CNF-SAT to Family Intersection

An example for a 2-CNF-SAT formula $\varphi = (\alpha \vee \beta) \wedge (\neg\alpha \vee \gamma)$:

- We number all the occurrences $(\alpha_1 \vee \beta_2) \wedge (\neg\alpha_3 \vee \gamma_4)$
- We interpret 4-bit numbers as the valuations of the occurrences

	False	True
α:	0000_2	1010_2
β:	0000_2	0100_2
γ:	0000_2	0001_2

	1st	2nd	1st and 2nd
$\alpha_1 \vee \beta_2$:	1000_2	0100_2	1100_2
$\neg\alpha_3 \vee \gamma_4$:	0000_2	0011_2	0001_2

- Left matrix represents consistent valuations of the occurrences.
- Right matrix represents valuations of the occurrences such that every clause is satisfied.

From Family Intersection to Common Matching Weight

From Family Intersection to Common Matching Weight

For a matrix with n rows and c columns we can build a weighted full bipartite graph such that:

- set of weights of all perfect matchings = set of all possible sums of choices
- number of the vertices is $O\left(\frac{n}{\log n}\right)$ (for fixed c)

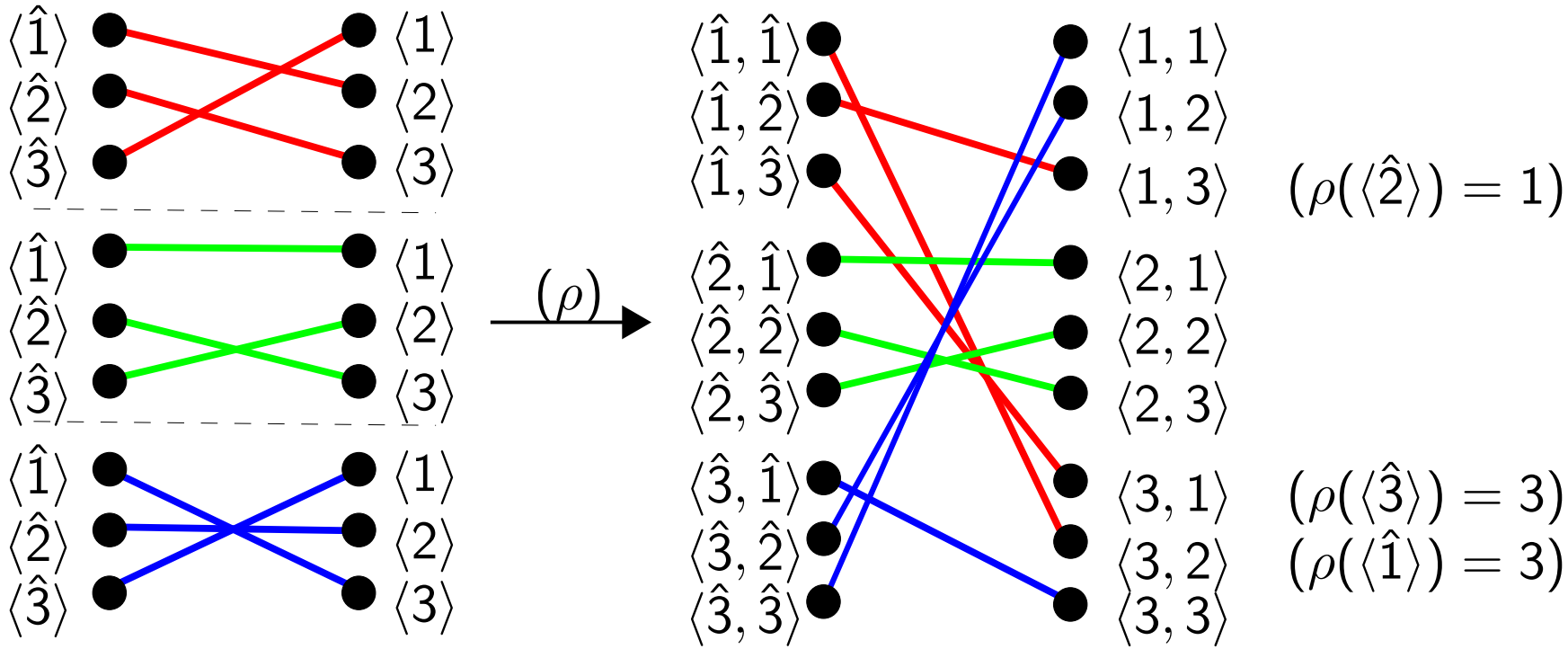
Common Matching Weight:

- Input: We are given two weighted bipartite graphs.
- Question: Is it possible to pick two perfect matchings in such a way that both sum of weights are equal?

From Family Intersection to Common Matching Weight

How do we compress? A nice recursive idea behind:

- Assume that we already have matchings that represents "something".
- By merging them we can additionally represent a function ρ on the left vertices of the first matching.



From Family Intersection to Common Matching Weight

We can use this idea to build our weighted graph recursively. When merging bipartite graphs:

- We copy the weights that for every vertex of the left side there is no difference to which of the merged parts it will be matched.
- Then for every vertex of the first part we can add to the weights of its edges a function depending only on the part to which it is connected.

From Family Intersection to Common Matching Weight

For the matrix with 4 rows and 2 columns we get:

$$[0]\ [0]\ [0]\ [0]$$

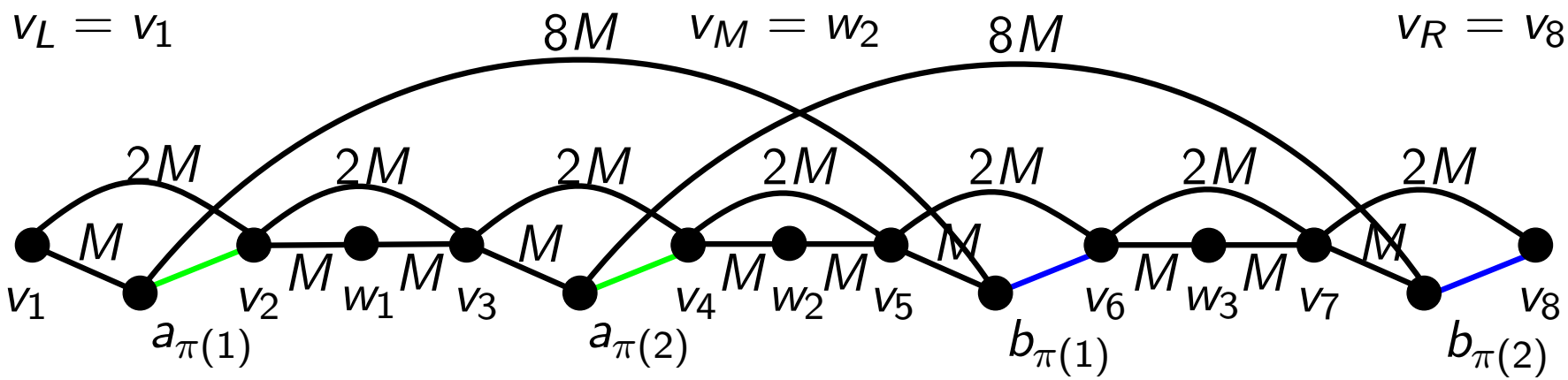
$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} A_{2,1} & A_{2,2} \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} A_{1,1}+A_{3,1} & A_{1,2}+A_{3,1} & A_{1,1}+A_{3,2} & A_{1,2}+A_{3,2} \\ A_{4,1} & A_{4,1} & A_{4,2} & A_{4,2} \\ A_{2,1} & A_{2,2} & A_{2,1} & A_{1,2} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

From Common Matching Weight to Channel Assignment

From Common Matching Weight to Channel Assignment

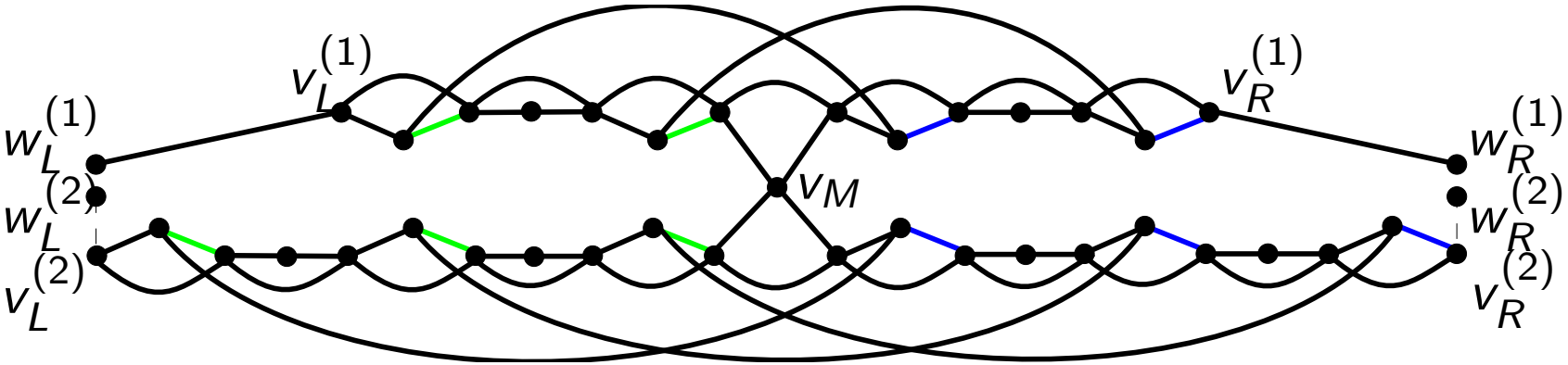
We encode a bipartite graph and its matchings into:



The position (color) of the middle vertex represent the weight of the matching corresponding to the permutation π.

From Common Matching Weight to Channel Assignment

We can encode two bipartite graphs with the common middle vertex. Its position (color) needs to represent the common weight of both matchings.



Reductions

$$\begin{array}{c}\text{3-CNF-}\\ \text{SAT}\\ O(n)\end{array} \rightarrow \begin{array}{c}\text{Family In-}\\ \text{tersection}\\ O(n)\end{array} \rightarrow \begin{array}{c}\text{Common}\\ \text{Matching}\\ \text{Weight}\\ O\left(\frac{n}{\log n}\right)\end{array} \rightarrow \begin{array}{c}\text{Channel}\\ \text{Assignment}\\ O\left(\frac{n}{\log n}\right)\end{array}$$

Summary

Our results

- Channel Assignment cannot be solved in $2^{o(n \log n)}$ under ETH (tight!)
- nor in $2^{n \cdot o(\log \log \ell)}$

Further research

- Faster algorithm for ℓ-bounded version? e.g. $O(\log \ell)^n$?

The end.