Total coloring of cubic graphs

Mathieu Vavrille

ENS de Lyon
M1 internship supervised by Łukasz Kowalik
in the University of Warsaw

5 September 2018

Introduction

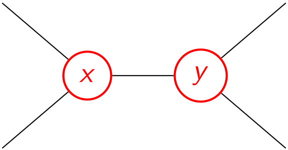
k-total coloring

Let $G = (V, E)$ a graph. A k-total coloring of G is a function $\mathcal{C} : V \cup E \to \{1, \ldots, k\}$ such that every two neighbouring elements have different colors.

Neighbouring element

u and $v \in V \cup E$ are two neighbouring elements iff:

- $u, v \in V$ and $(u, v) \in E$ (adjacent vertices)
- $u \in V, v = (u, w) \in E$ (vertex and incident edge)
- $u = (x, y), v = (x, w) \in E$ (incident edges)



Introduction

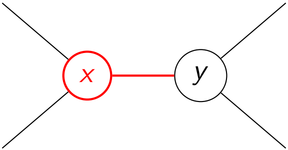
k-total coloring

Let $G = (V, E)$ a graph. A k-total coloring of G is a function $\mathcal{C} : V \cup E \to \{1, \ldots, k\}$ such that every two neighbouring elements have different colors.

Neighbouring element

u and $v \in V \cup E$ are two neighbouring elements iff:

- $u, v \in V$ and $(u, v) \in E$ (adjacent vertices)
- $u \in V, v = (u, w) \in E$ (vertex and incident edge)
- $u = (x, y), v = (x, w) \in E$ (incident edges)



Introduction

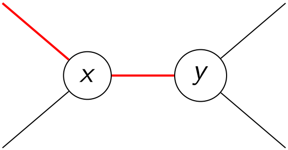
k-total coloring

Let $G = (V, E)$ a graph. A k-total coloring of G is a function $\mathcal{C} : V \cup E \to \{1, \ldots, k\}$ such that every two neighbouring elements have different colors.

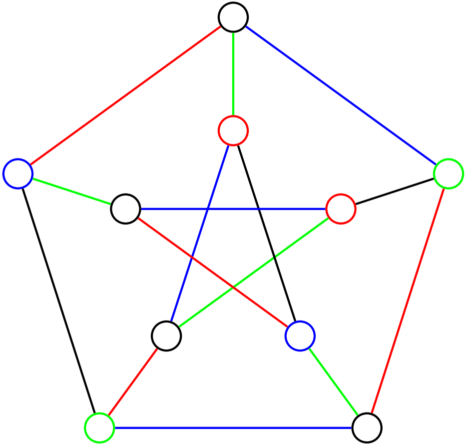
Neighbouring element

u and $v \in V \cup E$ are two neighbouring elements iff:

- $u, v \in V$ and $(u, v) \in E$ (adjacent vertices)
- $u \in V, v = (u, w) \in E$ (vertex and incident edge)
- $u = (x, y), v = (x, w) \in E$ (incident edges)



Example of coloring



Complexity

Three different problems

- Decision: deciding if a graph is k-totally colorable
- Counting: counting the number of coloring
- Enumeration: printing all the coloring

Complexity

Three different problems

- Decision: deciding if a graph is k-totally colorable
- Counting: counting the number of coloring
- Enumeration: printing all the coloring

NP-completeness of the decision problem [Sánchez-Arroyo, 1989]

$(\Delta(G)+1)$-total coloring is **NP**-complete, even on bipartite cubic graphs

Naive approach for the decision problem

Color the adjacency graph. Time complexity of $\mathcal{O}\left(2^{(\Delta+2)n/2}\right)$. Some improvements for small number of colors.

Prevous results

Exponential space for counting problem [Golovach et al., 2010]

Running time $\mathcal{O}\left(12^{(1/6+\epsilon)n}\right), \forall\epsilon > 0$, (bounded by $\mathcal{O}\left(2^{0.5975n}\right)$). Using dynamic programming over a path decomposition.

Polynomial space for enumeration problem [Bessy and Havet, 2013]

Running time $\mathcal{O}^*\left(2^{3n/2}\right)$, using an (s,t)-ordering of the graph.

Result of the internship

Running time $\mathcal{O}\left(2^{1.1943n}\right)$ and polynomial space.

Results of the internship

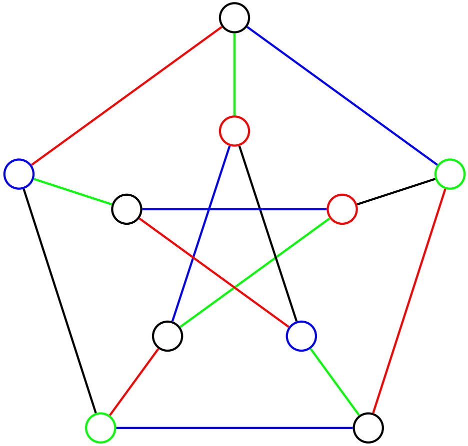
First approach

- Using a polynomial bipartitness check algorithm
- Analyzed with measure and conquer
- Running time $\mathcal{O}\left(2^{1.1943n}\right)$, and polynomial space.

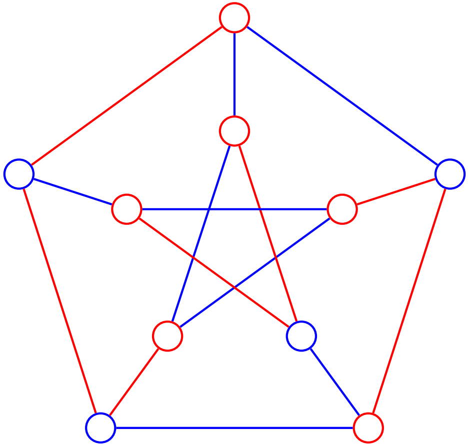
Second approach

- Using a polynomial total-list coloring algorithm on outerplanar graphs
- Analyzed with a linear program
- Running time $\mathcal{O}\left(2^{1.2893n}\right)$, and polynomial space.

The main lemma



The main lemma



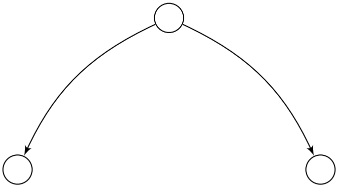
The main lemma

Lemma

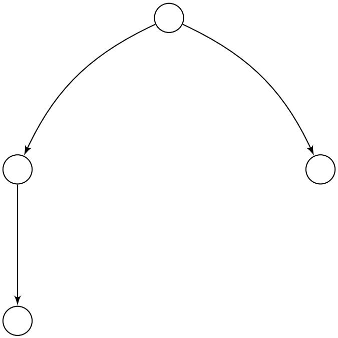
Let G be a cubic graph. If G it totally 4-colorable, then $E(G)$ can be partitioned into to subsets E_R (red edges) and E_B (blue edges) so that both E_R and E_B are collections of paths of length at least two, and even cycles.

Branching algorithms

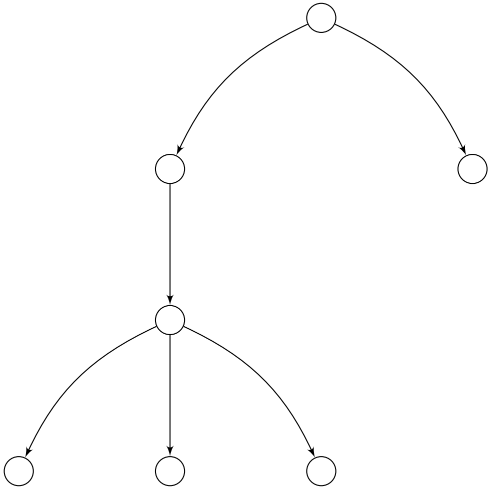
Branching algorithms



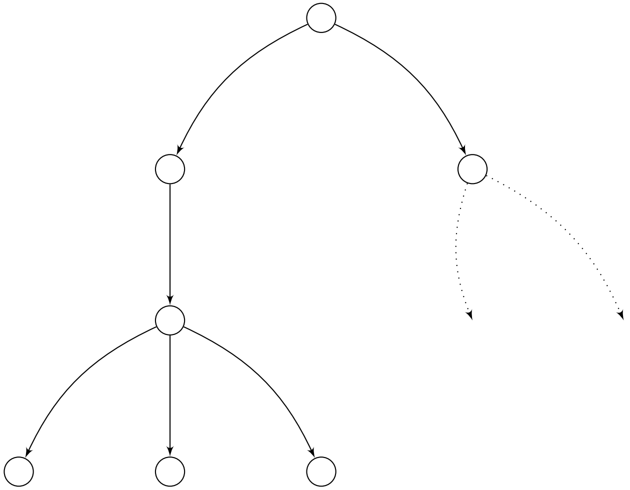
Branching algorithms



Branching algorithms



Branching algorithms



Different cases

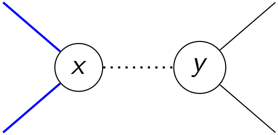


Figure 1: Case of reduction

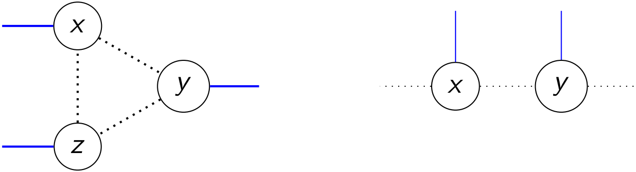


Figure 2: Technical edges

The algorithm

Algorithm 1 $\textsc{BipartiteSplit}(G_0, G, E_R, E_B)$

Input: The initial cubic graph G_0, the current red-blue decomposition E_R, E_B, and the remaining graph $G = (V, E)$
Output: True if G is totally 4-colorable, False otherwise
if $\exists xy \in E$ s.t. x is incident to two edges colored with the same color, say blue **then**
  return $\textsc{BipartiteSplit}(G_0, G - xy, E_R + xy, E_B)$
else if $\exists xy \in E$ s.t. xy is not technical **then**
  return $\begin{cases} \quad \textsc{BipartiteSplit}(G_0, G - xy, E_R + xy, & E_B) \\ \vee \; \textsc{BipartiteSplit}(G_0, G - xy, E_R, & E_B + xy) \end{cases}$
else if $\exists xy, yz, zx \in E$ s.t. x, y and z are touched by edges colored by the same color, say blue **then** ▷ It is a triangle
  return $\begin{cases} \quad \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{wx, yz\} & , E_B + xy) \\ \vee \; \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx\} & , E_B + yz) \\ \vee \; \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, yz\} & , E_B + wx) \end{cases}$
else if $\exists wx, xy, yz \in E$ s.t. x and y are incident to edges colored by the same color, say blue **then**
  return $\begin{cases} \quad \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{wx, yz\} & , E_B + xy) \\ \vee \; \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx, yz\} & , E_B) \\ \vee \; \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx\} & , E_B + yz) \\ \vee \; \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, yz\} & , E_B + wx) \end{cases}$
else ▷ Every edge is colored
  $\textsc{CheckBipartite}(G, E_B) \wedge \textsc{CheckBipartite}(G, E_R)$
end if

The algorithm

Algorithm 2 $\textsc{BipartiteSplit}(G_0, G, E_R, E_B)$

Input: The initial cubic graph G_0, the current red-blue decomposition E_R, E_B, and the remaining graph $G = (V, E)$
Output: True if G is totally 4-colorable, False otherwise
if $\exists xy \in E$ s.t. x is incident to two edges colored with the same color, say blue **then**
  return $\textsc{BipartiteSplit}(G_0, G - xy, E_R + xy, E_B)$
else if $\exists xy \in E$ s.t. xy is not technical **then**
  return $\begin{cases} \quad \textsc{BipartiteSplit}(G_0, G - xy, E_R + xy, \quad E_B) \\ \vee \ \textsc{BipartiteSplit}(G_0, G - xy, E_R, \qquad E_B + xy) \end{cases}$
else if $\exists xy, yz, zx \in E$ s.t. x, y and z are touched by edges colored by the same color, say blue **then** ▷ It is a triangle
  return $\begin{cases} \quad \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{wx, yz\}, E_B + xy) \\ \vee \ \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx\}, E_B + yz) \\ \vee \ \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, yz\}, E_B + wx) \end{cases}$
else if $\exists wx, xy, yz \in E$ s.t. x and y are incident to edges colored by the same color, say blue **then**
  return $\begin{cases} \quad \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{wx, yz\}, E_B + xy) \\ \vee \ \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx, yz\}, E_B) \\ \vee \ \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx\}, E_B + yz) \\ \vee \ \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, yz\}, E_B + wx) \end{cases}$
else ▷ Every edge is colored
  $\textsc{CheckBipartite}(G, E_B) \wedge \textsc{CheckBipartite}(G, E_R)$
end if

The algorithm

Algorithm 3 $\textsc{BipartiteSplit}(G_0, G, E_R, E_B)$

Input: The initial cubic graph G_0, the current red-blue decomposition E_R, E_B, and the remaining graph $G = (V, E)$
Output: True if G is totally 4-colorable, False otherwise
if $\exists xy \in E$ s.t. x is incident to two edges colored with the same color, say blue **then**
    return $\textsc{BipartiteSplit}(G_0, G - xy, E_R + xy, E_B)$
else if $\exists xy \in E$ s.t. xy is not technical **then**
    return $\begin{cases} & \textsc{BipartiteSplit}(G_0, G - xy, E_R + xy, \quad E_B) \\ \vee & \textsc{BipartiteSplit}(G_0, G - xy, E_R, \qquad E_B + xy) \end{cases}$
else if $\exists xy, yz, zx \in E$ s.t. x, y and z are touched by edges colored by the same color, say blue **then** ▷ It is a triangle
    return $\begin{cases} & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{wx, yz\} \quad, E_B + xy) \\ \vee & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx\} \quad, E_B + yz) \\ \vee & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, yz\} \quad, E_B + wx) \end{cases}$
else if $\exists wx, xy, yz \in E$ s.t. x and y are incident to edges colored by the same color, say blue **then**
    return $\begin{cases} & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{wx, yz\} \qquad, E_B + xy) \\ \vee & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx, yz\} \quad, E_B) \\ \vee & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, wx\} \qquad, E_B + yz) \\ \vee & \textsc{BipartiteSplit}(G_0, G - \{xy, yz, wx\}, E_R + \{xy, yz\} \qquad, E_B + wx) \end{cases}$
else ▷ Every edge is colored
    $\textsc{CheckBipartite}(G, E_B) \wedge \textsc{CheckBipartite}(G, E_R)$
end if

Analysis of branching algorithms

Reductions and Branchings

There are two sets of recursive calls:

- Reduction rules, with only one recursive call
- Branching rules, with more than one

We focus on branching rules

Solving recurrence

For each recurrence $T(s) = T(s - a_1) + T(s - a_2)$ we compute the maximum zero $\lambda(a_1, a_2)$ of the function $X \mapsto 1 - X^{-a_1} - X^{-a_2}$. Then $T(s) = \mathcal{O}^*\left(\lambda^s\right)$

Measure and conquer [Fomin et al., 2009]

Idea

- Define a different size of the instance
- Put weights on the different cases
- Express the recurrences as a function of these weights
- Tune the weights

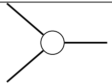
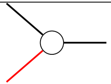
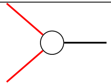
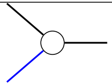
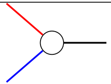
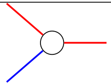
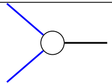
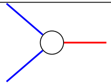
$a_{x,y}$	$x = 0$	$x = 1$	$x = 2$
$y = 0$			
$y = 1$			
$y = 2$			

Table 1: Table linking the vertices to their weight

Example of recurrences

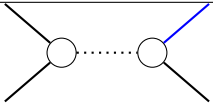
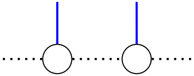
illustration	symmetric ?	size of recursive calls
	Yes	$s - a_{0,0} - \min(a_{x,y} - a_{x+1,y})$ $s - a_{0,0} - a_{0,1} + a_{1,0} + a_{1,1}$
	Yes	$s - 4a_{0,1} + 2a_{1,1}$ $s - 4a_{0,1} + 2a_{1,1}$ $s - 4a_{0,1} + a_{1,1}$ $s - 4a_{0,1} + a_{1,1}$

Table 2: Two examples of recurrences

Detailed example (1)

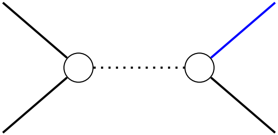
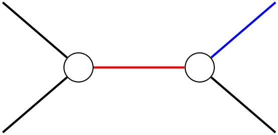


Figure 3: Case to treat



$$s - a_{0,0} - a_{0,1} + a_{1,0} + a_{1,1}$$

Detailed example (1)

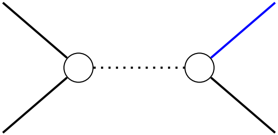
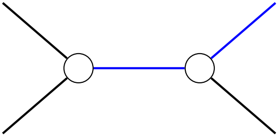


Figure 3: Case to treat



$s - a_{0,0}$

Detailed example (1)

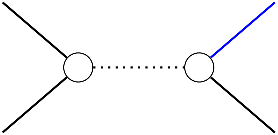
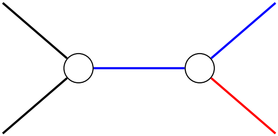


Figure 3: Case to treat



$$s - a_{0,0} - \min(a_{x,y} - a_{x+1,y})$$

Detailed example (2)

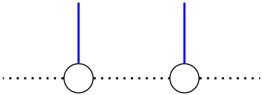
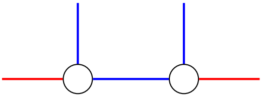


Figure 4: Case to treat



$$s - 4a_{0,1} - 2a_{1,1}$$

Detailed example (2)

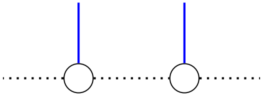
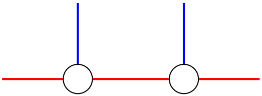


Figure 4: Case to treat



$$s - 4a_{0,1} - 2a_{1,1}$$

Detailed example (2)

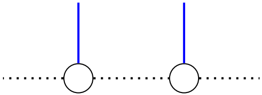
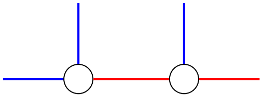


Figure 4: Case to treat



$$s - 4a_{0,1} - a_{1,1}$$

Detailed example (2)

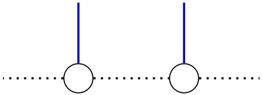
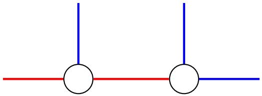
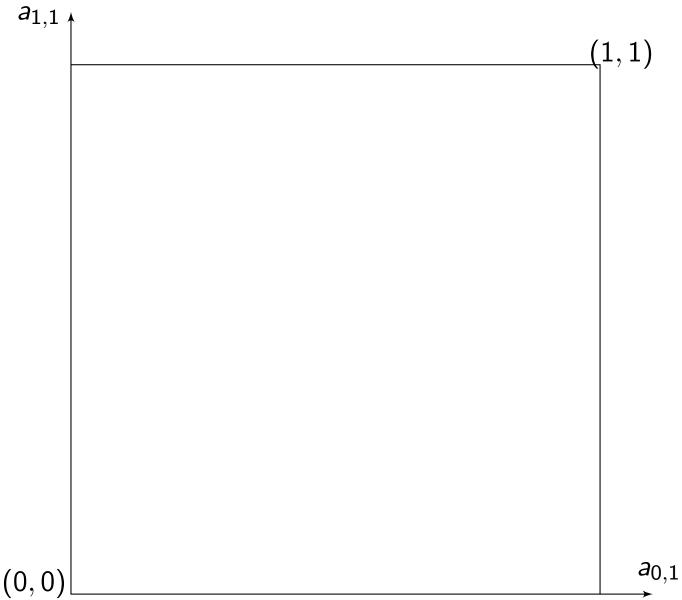


Figure 4: Case to treat

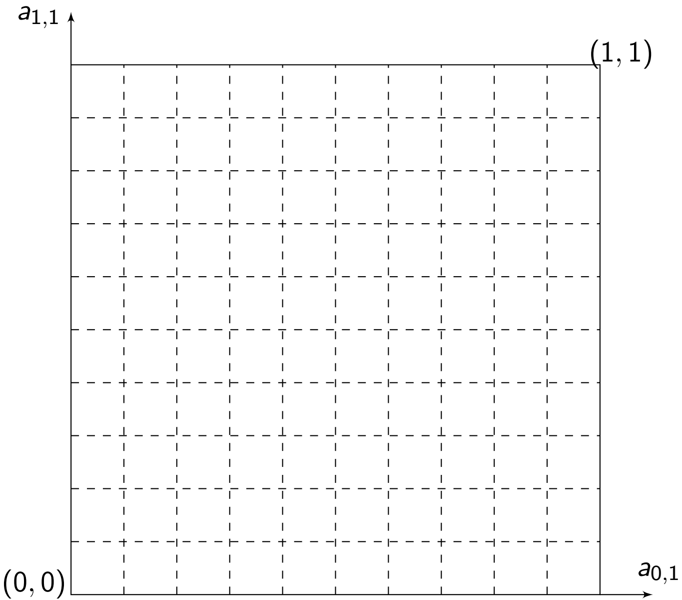


$$s - 4a_{0,1} - a_{1,1}$$

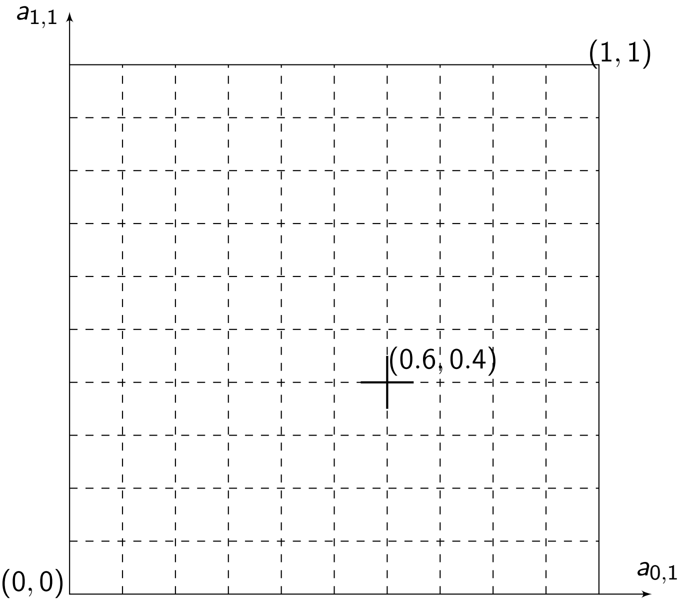
Finding good coefficients



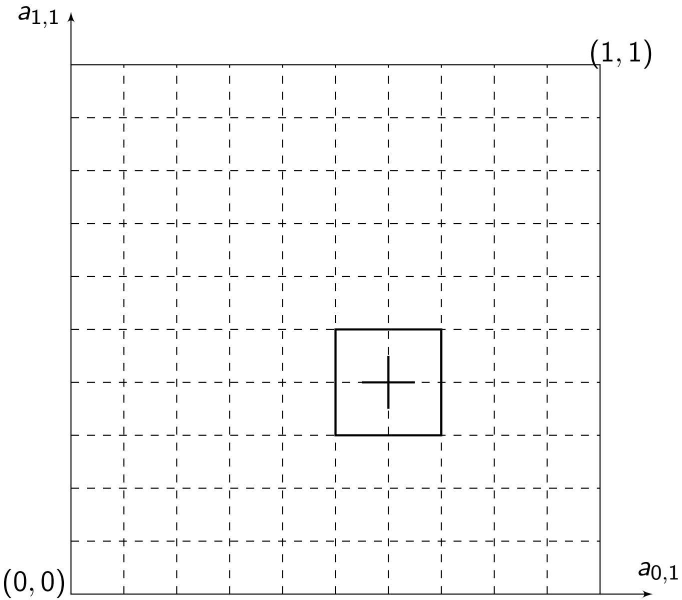
Finding good coefficients



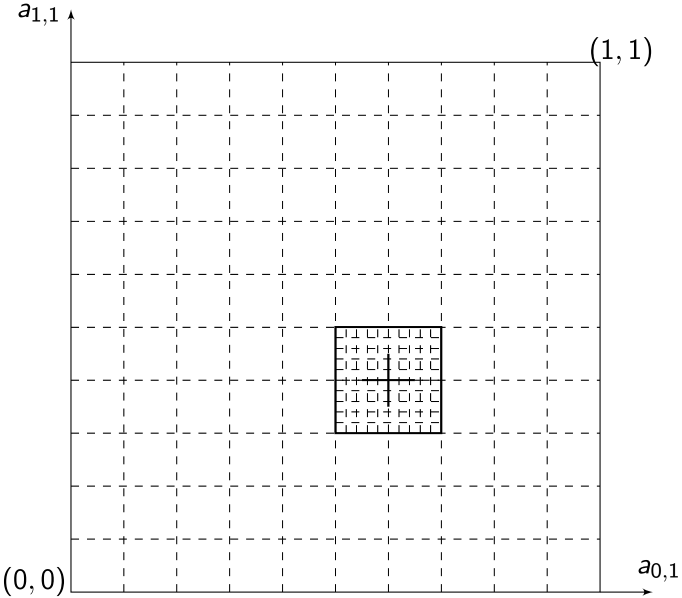
Finding good coefficients



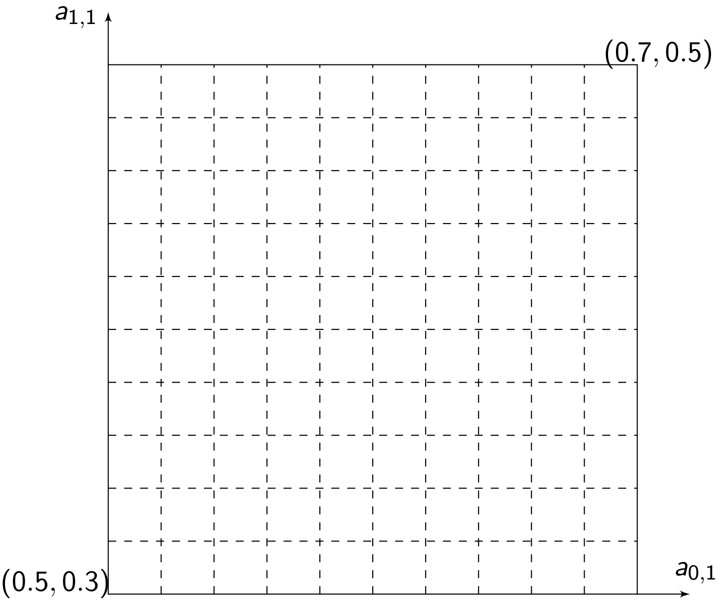
Finding good coefficients



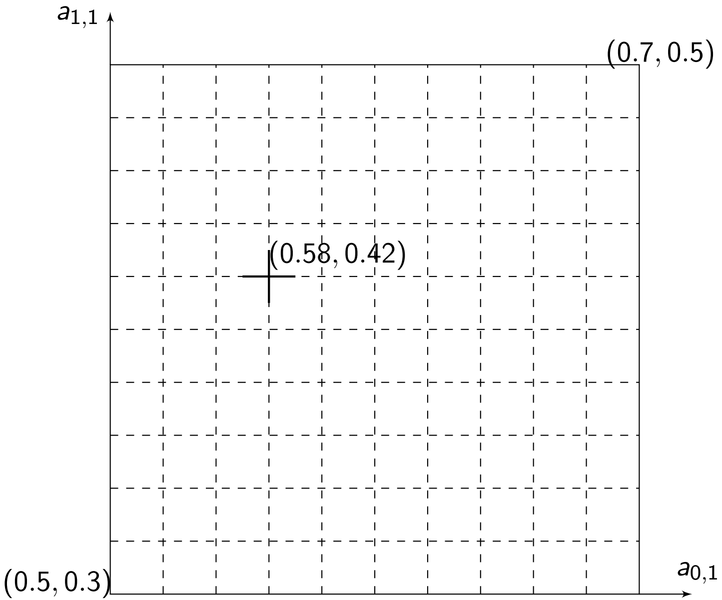
Finding good coefficients



Finding good coefficients



Finding good coefficients



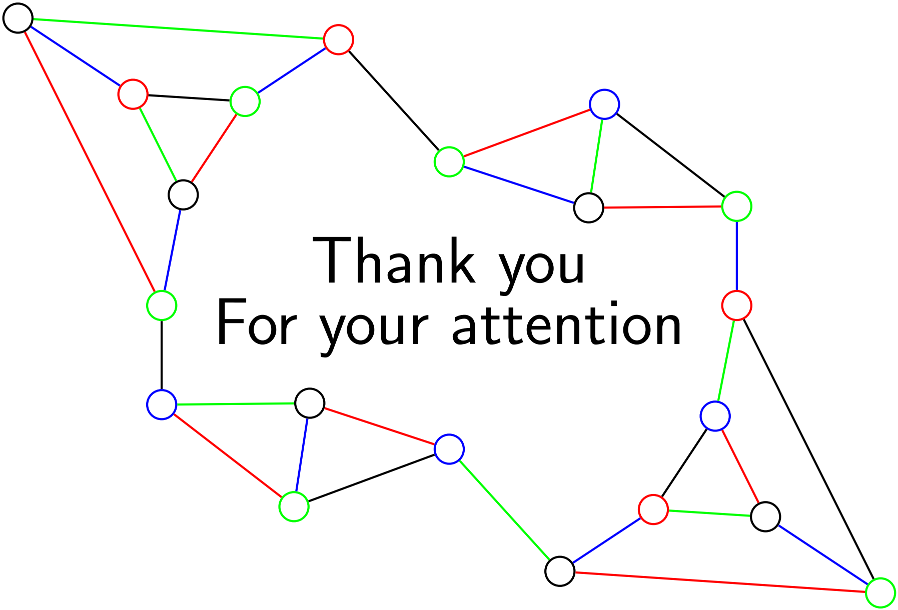
Results

Result of the script

- Weights: $a_{0,1} = 0.5813, a_{1,1} = 0.4187$
- Work factor: $\lambda = 2.28825$ (so the complexity is $\mathcal{O}(\lambda^n)$)

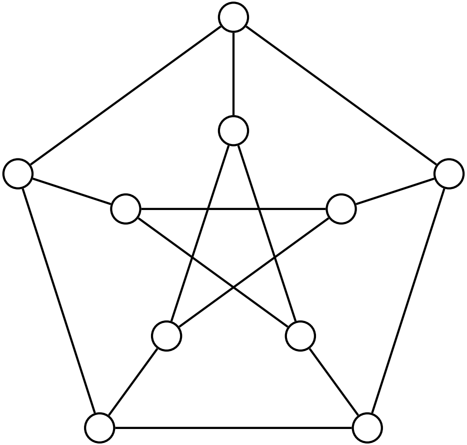
Complexity

- Running time $\mathcal{O}\left(2^{1.1943n}\right)$ and polynomial space
- Solving decision problem
- Can be extended to counting problem

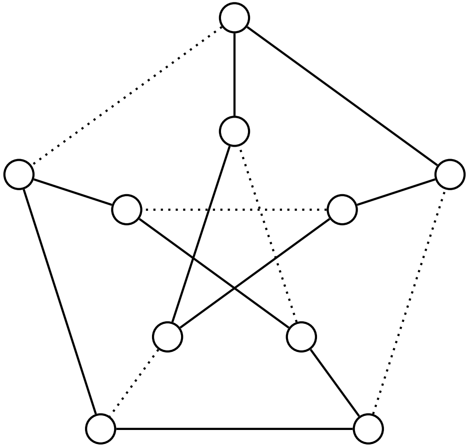


Thank you
For your attention

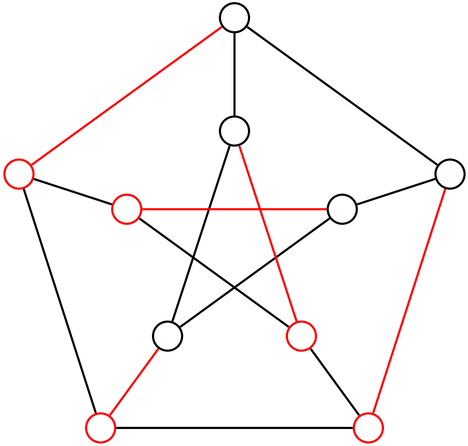
Idea on an example



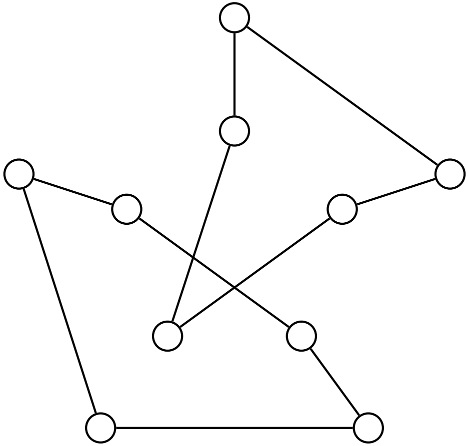
Idea on an example



Idea on an example



Idea on an example



Idea

Algorithm

- Find a matching touching all degree three vertices
- Color the edges of the matching and one incident vertex in all possible way
- Check total-list colorability of the remaining graph

Improvements

- Find restrictions when coloring the edges and vertices
- Don't color all the matched edges

Remaining graph

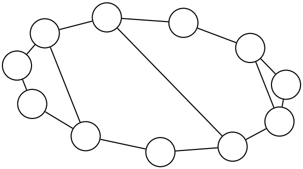


Figure 5: An example of outerplanar graph

De Courcelle's theorem

Every graph property definable in the monadic second-order logic of graphs can be decided in linear time on graphs of bounded treewidth.

Remaining graph

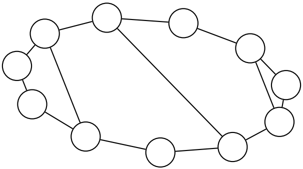


Figure 5: An example of outerplanar graph

De Courcelle's theorem

Every graph property definable in the monadic second-order logic of graphs can be decided in linear time on graphs of bounded treewidth.

Properties we have

- Outerplanar graphs have bounded treewidth
- The total-list coloring problem can be defined in the monadic second-order logic

Analysis

Using a linear program

- For each case of branching, create a coefficient a_i
- Constraints on a_i express non-trivial restrictions of the algorithm
- Maximization function express the exponent of the complexity, known using the number of branching of case a_i.

Bessy, S. and Havet, F. (2013).
Enumerating the edge-colourings and total colourings of a regular graph.
Journal of Combinatorial Optimization, 25(4):523–535.

Fomin, F. V., Grandoni, F., and Kratsch, D. (2009).
A measure & conquer approach for the analysis of exact algorithms.
Journal of the ACM (JACM), 56(5):25.

Golovach, P. A., Kratsch, D., and Couturier, J.-F. (2010).
Colorings with few colors: counting, enumeration and combinatorial bounds.
In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 39–50. Springer.

Sánchez-Arroyo, A. (1989).
Determining the total colouring number is np-hard.
Discrete Mathematics, 78(3):315–319.