Fast Exact Algorithm for $L(2,1)$-Labeling of Graphs

Mathieu Liedloff

Université d'Orléans - LIFO

joint work with:

Konstanty Junosza-Szaniawski [1] Jan Kratochvíl [2]

Peter Rossmanith [3] Paweł Rzążewski [1]

[1] Warsaw University of Technology,
Faculty of Mathematics and Information Science,
Warszawa, Poland

[2] Department of Applied Mathematics,
and Institute for Theoretical Computer Science, Charles University,
Praha, Czech Republic

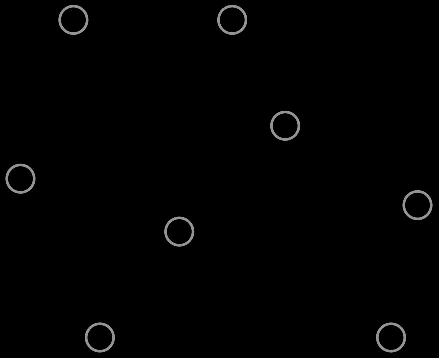
[3] Department of Computer Science, RWTH Aachen University,
Aachen, Germany

Journées Franciliennes de Recherche Opérationnelle
20 novembre 2012

Outline

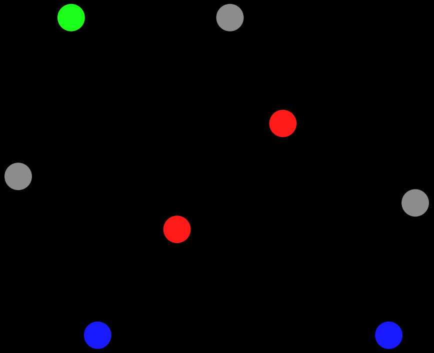
Frequency assignment problem

- broadcast network
- assign frequencies to transmitters
- avoid undesired interference



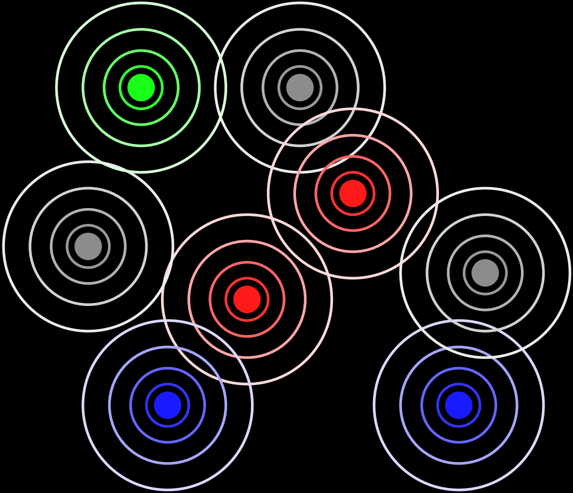
Frequency assignment problem

- broadcast network
- assign frequencies to transmitters
- avoid undesired interference



Frequency assignment problem

- broadcast network
- assign frequencies to transmitters
- avoid undesired interference

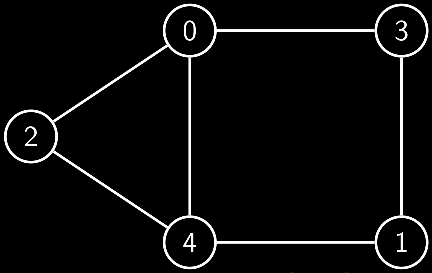


Definition of $L(2,1)$-labeling

$L(2,1)$-LABELING

Input : A graph $G = (V, E)$.
Question : Compute a function ℓ of minimum span k
$\ell : V \to \{0, \ldots, k\}$ s.t.

- u and v adjacent $\Rightarrow |\ell(u) - \ell(v)| \geq 2$
- u and v at distance two $\Rightarrow |\ell(u) - \ell(v)| \geq 1$



$\to$ Model introduced by Roberts, 1988 [Rob].

Known complexity results

Theorem **[GY92]**

Determining the minimum span $\lambda(G)$ of a graph G is NP-hard.

Theorem **[FKK01]**

Deciding whether $\lambda(G) \leq k$ remains NP-complete for every fixed $k \geq 4$. (trivial for $k \leq 3$)

Theorem **[CK96, FGK05]**

When the span k is part of the input, $L(2,1)$-labeling problem is polynomial time solvable on trees.

However, the problem is NP-complete for series-parallel graphs.

$\rightarrow$ The problem "separates" graphs of treewidth 1 and 2 by P / NP-completeness dichotomy.

Known complexity results

Theorem **[GY92]**

Determining the minimum span $\lambda(G)$ of a graph G is NP-hard.

Theorem **[FKK01]**

Deciding whether $\lambda(G) \leq k$ remains NP-complete for every fixed $k \geq 4$. (trivial for $k \leq 3$)

Theorem **[CK96, FGK05]**

When the span k is part of the input,
$L(2,1)$-labeling problem is polynomial time solvable on trees.

However, the problem is NP-complete for series-parallel graphs.

$\rightarrow$ The problem "separates" graphs of treewidth 1 and 2 by P / NP-completeness dichotomy.

Known complexity results

The distance constrained labeling problem is more difficult than ordinary coloring :

Theorem **[FGK05]**

Deciding whether $\lambda(G) \leq k$ is NP-complete for series-parallel graphs (k is part of the input).

Theorem **[BKTvL04, JKM09]**

Deciding whether $\lambda = k$ is NP-complete for planar graphs

- for $k = 8$ [BKTvL04]
- for $k = 4$ [JKM09]

$L(2,1)$-labeling and Locally Injective Homomorphisms

Fiala and Kratochvíl defined the notion of $H(2,1)$-labeling :

- mapping from vertices of G to vertices of a graph H ;
- adjacent vertices in G are mapped onto non-adjacent vertices in H ;
- vertices with a common neighbor in G are mapped onto distinct vertices of H.

They show that :

$\rightarrow$ $H(2,1)$-labelings are exactly locally injective homomorphisms from G to $\overline{H}$.

$\rightarrow$ $L(2,1)$-labeling of span k is a locally injective homomorphism into the complement of the path of length k.

$L(2,1)$-labeling and Locally Injective Homomorphisms

Fiala and Kratochvíl defined the notion of $H(2,1)$-labeling :

- mapping from vertices of G to vertices of a graph H ;
- adjacent vertices in G are mapped onto non-adjacent vertices in H ;
- vertices with a common neighbor in G are mapped onto distinct vertices of H.

They show that :

$\rightarrow$ $H(2,1)$-labelings are exactly locally injective homomorphisms from G to $\overline{H}$.

$\rightarrow$ $L(2,1)$-labeling of span k is a locally injective homomorphism into the complement of the path of length k.

$L(2,1)$-labeling and Locally Injective Homomorphisms

Fiala and Kratochvíl defined the notion of $H(2,1)$-labeling :

- mapping from vertices of G to vertices of a graph H ;
- adjacent vertices in G are mapped onto non-adjacent vertices in H ;
- vertices with a common neighbor in G are mapped onto distinct vertices of H.

They show that :

$\rightarrow$ $H(2,1)$-labelings are exactly locally injective homomorphisms from G to $\overline{H}$.

$\rightarrow$ $L(2,1)$-labeling of span k is a locally injective homomorphism into the complement of the path of length k.

$L(2,1)$-labeling and Locally Injective Homomorphisms

homomorphism : A mapping $f : V(G) \to V(H)$ is a homomorphism from G to H if $f(u)f(v) \in E(H)$ for every edge $uv \in E(G)$.

Theorem **[HN90]**

Homomorphisms admit a complete dichotomy :
Deciding existence of a homomorphism into a fixed graph H is

- polynomial when H is bipartite ;
- NP-complete otherwise.

Remark : k-coloring of a graph G corresponds to homomorphism from G to the graph K_k.

$L(2,1)$-labeling and Locally Injective Homomorphisms

homomorphism : A mapping $f : V(G) \to V(H)$ is a homomorphism from G to H if $f(u)f(v) \in E(H)$ for every edge $uv \in E(G)$.

locally injective homomorphism (LIH) : A homomorphism $f : G \to H$ is locally injective if for every vertex $u \in V(G)$ its neighborhood is mapped injectively into the neighborhood of $f(u)$ in H, i.e., every two vertices having a common neighbor in G are mapped onto disctinct vertices in H.

Theorem **[HKKKL11]**

H-locally-injective-homomorphism can be solved in time

$$O^*\left((\Delta(H) - 1)^n\right)$$

$L(2,1)$-labeling and Locally Injective Homomorphisms

homomorphism : A mapping $f : V(G) \to V(H)$ is a homomorphism from G to H if $f(u)f(v) \in E(H)$ for every edge $uv \in E(G)$.

locally injective homomorphism (LIH) : A homomorphism $f : G \to H$ is locally injective if for every vertex $u \in V(G)$ its neighborhood is mapped injectively into the neighborhood of $f(u)$ in H, i.e., every two vertices having a common neighbor in G are mapped onto disctinct vertices in H.

Theorem **[HKKKL11]**

H-locally-injective-homomorphism can be solved in time

$$O^*\left((\Delta(H)-1)^n\right)$$

$L(2,1)$-labeling problem - Exact algorithms

Theorem **[HKKKL11]**

H-locally-injective-homorphism can be solved in time

$$O^*\left((\Delta(H)-1)^n\right)$$

$\rightarrow$ $L(2,1)$-labeling of span k is a locally injective homomorphism into the complement of the path of length k.

Theorem **[HKKKL11]**

Hence, $L(2,1)$-labeling problem of span k can be decided in time

$$O^*\left((k-1)^n\right)$$

$L(2,1)$-labeling problem - Exact algorithms

Theorem **[HKKKL11]**
$L(2,1)$-labeling of span 4 : $O(1.3006^n)$ *(branching)*

Theorem **[GKC10]**
$L(2,1)$-labeling of span 5 in cubic graphs : $O(1.8613^n) \to O(1.7990^n)$

Theorem **[Král'06]**
$L(2,1)$-labeling of min span : $O^*(4^n)$

Theorem **[HKKKL11]**
$L(2,1)$-labeling of min span : $O^*(15^{n/2}) = O(3.88^n)$ *(D.P.)*

Theorem **[HKKKL08], [J-SKLR12]**
$L(2,1)$-labeling of min span : $O((9+\epsilon)^n) \to O(7.50^n)$ *(D. & C.)*

Theorem **[CK11]**
$L(2,1)$-labeling of min span : $O^*(3^n)$ *(fast ζ transform + I.-E.)*

Can the problem be solved faster ?

A DP based algorithm for $L(2,1)$-labeling of min span

A DP based algorithm for $L(2,1)$-labeling of min span

How to compute an $L(2,1)$-labeling of span k by Dynamic Programming ?

First, we show the following :

Theorem :
An $L(2,1)$)labeling of span k can be decided in time $O^*(4^n)$.

A DP based algorithm for $L(2,1)$-labeling of min span

How to compute an $L(2,1)$-labeling of span k by Dynamic Programming ?

First, we show the following :

Theorem :
An $L(2,1)$)labeling of span k can be decided in time $O^*(4^n)$.

bound on the number of 2-packings

bound on the number of 2-packings

Theorem :
An $L(2,1)$)labeling of span k can be decided in time $O^*(3.88^n)$.

2-**packings** $=$ **Independent Sets in** G^2
A subset $S \subseteq V$ s.t. $\forall u, v \in S$, $N[u] \cap N[v] = \emptyset$ is a 2-packing.

(2-packing $\equiv$ set of vertices pairwise at distance greater than 2.)

A DP based algorithm for $L(2,1)$-labeling of min span

Reminder :
Let $G = (V, E)$ be a graph. An $L(2,1)$-labeling of span k asks to find a labeling f of G such that :

- for all $\{u, v\} \in E \quad \Rightarrow \quad |f(u) - f(v)| \geq 2$;
- for all $u, v \in V$ s.t. $dist(u, v) = 2 \quad \Rightarrow \quad f(u) \neq f(v)$.

$\forall i \in \{0, 1, \ldots, k\}$ and $\forall X, Y \subseteq V$ such that $X \cap Y = \emptyset$, we define the boolean variable $\texttt{Lab}(X, Y, i)$.

$\texttt{Lab}(X, Y, i)$ is $\texttt{true}$ iff
there is an $L(2,1)$-labeling of span i of the vertices of X such that the vertices of $N(Y) \cap X$ have label at most $i - 1$.

A DP based algorithm for $L(2,1)$-labeling of min span

Reminder : $\texttt{Lab}(X, Y, i)$ is $\texttt{true}$ iff
there is an $L(2,1)$-labeling of span i of the vertices of X such that
the vertices of $N(Y) \cap X$ have label at most $i-1$.

It is not difficult to check that

- $\texttt{Lab}(\emptyset, Y, i) \leftarrow \texttt{true} \quad \forall Y, \forall i;$
- $$\texttt{Lab}(X, Y, 0) \leftarrow \begin{cases} \texttt{true} & \forall X, Y \text{ s.t. } X \text{ is an indep. set of } G^2 \text{ and } X \cap N(Y) = \emptyset \\ \texttt{false} & \text{otherwise} \end{cases}$$

Then, $\texttt{Lab}(X, Y, i)$ is computed by considering the sets X and Y by increasing order of cardinality, and by increasing value of i :

$\texttt{Lab}(X, Y, i) = \texttt{true}$ iff $\quad \exists U \subseteq (X \setminus N(Y))$ such that

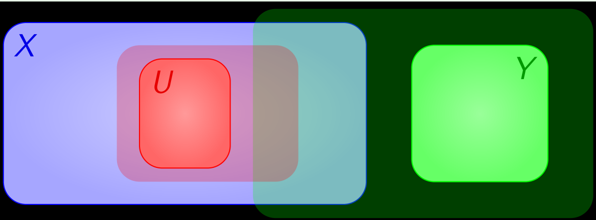
- U is a 2-packing of G ; and
- $\texttt{Lab}(X \setminus U, U, i-1) = \texttt{true}.$

A DP based algorithm for $L(2,1)$-labeling of min span

Reminder : $\texttt{Lab}(X, Y, i)$ is $\texttt{true}$ iff
there is an $L(2,1)$-labeling of span i of the vertices of X such that
the vertices of $N(Y) \cap X$ have label at most $i-1$.

Reminder : $\texttt{Lab}(X, Y, i) = \texttt{true}$ iff $\exists U \subseteq (X \setminus N(Y))$ such that

- U is a 2-packing of G ; and
- $\texttt{Lab}(X \setminus U, U, i-1) = \texttt{true}$.

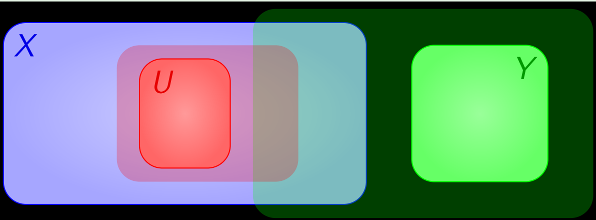


A DP based algorithm for $L(2,1)$-labeling of min span

Reminder : $\mathtt{Lab}(X, Y, i)$ is $\mathtt{true}$ iff
there is an $L(2,1)$-labeling of span i of the vertices of X such that
the vertices of $N(Y) \cap X$ have label at most $i-1$.

Reminder : $\mathtt{Lab}(X, Y, i) = \mathtt{true}$ iff $\exists U \subseteq (X \setminus N(Y))$ such that

- U is a 2-packing of G ; and
- $\mathtt{Lab}(X \setminus U, U, i-1) = \mathtt{true}$.



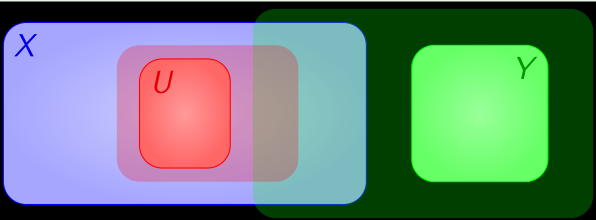
If X has an $L(2,1)$)-labeling of span i then
there is a (possibly empty) set $U \subseteq X \setminus N(Y)$ of vertices having label i. This set is a 2-packing of G.

A DP based algorithm for $L(2,1)$-labeling of min span

Reminder : $\texttt{Lab}(X, Y, i)$ is $\texttt{true}$ iff
there is an $L(2,1)$-labeling of span i of the vertices of X such that the vertices of $N(Y) \cap X$ have label at most $i-1$.

Reminder : $\texttt{Lab}(X, Y, i) = \texttt{true}$ iff $\exists U \subseteq (X \setminus N(Y))$ such that

- U is a 2-packing of G ; and
- $\texttt{Lab}(X \setminus U, U, i-1) = \texttt{true}$.



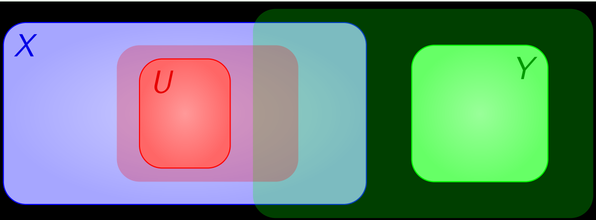
$\Rightarrow$ the neighbors of U must obtain label at most $i-2$ and $X \setminus U$ must have an $L(2,1)$-labeling of span at most $i-1$.
If a such labeling exists then $\texttt{Lab}(X \setminus U, U, i-1) = \texttt{true}$.

A DP based algorithm for $L(2,1)$-labeling of min span

Reminder : $\texttt{Lab}(X, Y, i)$ is $\texttt{true}$ iff
there is an $L(2,1)$-labeling of span i of the vertices of X such that
the vertices of $N(Y) \cap X$ have label at most $i-1$.

Reminder : $\texttt{Lab}(X, Y, i) = \texttt{true}$ iff $\exists U \subseteq (X \setminus N(Y))$ such that

- U is a 2-packing of G ; and
- $\texttt{Lab}(X \setminus U, U, i-1) = \texttt{true}$.



Remark : the vertices of $X \cap N(Y)$ in this labeling have label at most $i-1$.

A DP based algorithm for $L(2,1)$-labeling of min span

Running-time analysis :

$\texttt{Lab}(X, Y, i)$ is computed for all $X, Y \subseteq V$ such that $X \cap Y = \emptyset$, and for all $i \in \{0, 1, \dots, k\}$.

For each X, Y, we compute all sets $U \subseteq X$ being 2-packings of G.

$$k \cdot \sum_{x=0}^{n} \left(\binom{n}{x} \sum_{y=0}^{n-x} \binom{n-x}{y} \sum_{u=0}^{x} \binom{x}{u} \right)$$

A DP based algorithm for $L(2,1)$-labeling of min span

Running-time analysis :

$\texttt{Lab}(X, Y, i)$ is computed for all $X, Y \subseteq V$ such that $X \cap Y = \emptyset$, and for all $i \in \{0, 1, \ldots, k\}$.

For each X, Y, we compute all sets $U \subseteq X$ being 2-packings of G.

$$k \cdot \sum_{x=0}^{n} \left(\binom{n}{x} \sum_{y=0}^{n-x} \binom{n-x}{y} \sum_{u=0}^{x} \binom{x}{u} \right)$$

A DP based algorithm for $L(2,1)$-labeling of min span

Running-time analysis :

$\texttt{Lab}(X, Y, i)$ is computed for all $X, Y \subseteq V$ such that $X \cap Y = \emptyset$, and for all $i \in \{0, 1, \ldots, k\}$.

For each X, Y, we compute all sets $U \subseteq X$ being 2-packings of G.

$$
\begin{aligned}
& k \cdot \sum_{x=0}^{n} \left(\binom{n}{x} \sum_{y=0}^{n-x} \binom{n-x}{y} \sum_{u=0}^{x} \binom{x}{u} \right) \\
= \quad & k \cdot \sum_{x=0}^{n} \left(\binom{n}{x} 2^{n-x} 2^{x} \right) \\
= \quad & k \cdot 2^n \cdot 2^n
\end{aligned}
$$

Theorem :
Computing an $L(2,1)$ of span k can be obtain in time $O^*(4^n)$.

A DP based algorithm for $L(2,1)$-labeling of min span

By using a bound on the number of 2-packing of a certain size,

Theorem **[HKKKL11]**

Let u_k be the number of 2-packings of size k in a connected graph. Then,

$$u_k \leq \binom{n/2}{k} \cdot 2^k$$

$$u_k = 0 \text{ for } k > n/2$$

we are able to prove that :

Theorem :

An $L(2,1)$ of span k can be obtain in time $O^*(4^n) \rightsquigarrow O^*(3.8730^n)$.

[improving upon Král's result]

Note : These results can be extended to $L(p,q)$-labelings.

An auxiliary combinatorial result

2-Packings and Proper Pairs

Like *independent sets* are heavily related to colorings,
it seems that 2-*packings* are related to $L(2,1)$-labelings.

Theorem :
An $L(2,1)$ of span k can be obtain in time $O^*(2.6488^n)$.

But in fact we need another combinatorial object :

Proper Pairs

... and we need a bound on its maximum number in a graph.

2-Packings and Proper Pairs

Like *independent sets* are heavily related to colorings,
it seems that *2-packings* are related to $L(2,1)$-labelings.

Theorem :
An $L(2,1)$ of span k can be obtain in time $O^*(2.6488^n)$.

But in fact we need another combinatorial object :

Proper Pairs

... and we need a bound on its maximum number in a graph.

... and Proper Pairs

Definition

A pair (S, X) of subsets of V is a proper pair if $S \cap X = \emptyset$ and S is a 2-packing.

Definition

The number of proper pairs in a graph G is given by

$$pp(G) = \sum_{2\text{–packings } S} 2^{n-|S|}$$

Let $pp(n) = \max pp(G)$ be the maximum number of proper pairs in a connected graph with n vertices.

… and Proper Pairs

Definition

A pair (S, X) of subsets of V is a proper pair if $S \cap X = \emptyset$ and S is a 2-packing.

Definition

The number of proper pairs in a graph G is given by

$$pp(G) = \sum_{2-\text{packings } S} 2^{n-|S|}$$

Let $pp(n) = \max pp(G)$ be the maximum number of proper pairs in a connected graph with n vertices.

Theorem

$$2.6117^n \leq pp(n) \leq 2.6488^n$$

(will be very useful in the next)

... and Proper Pairs

Proof. **1/2**

Let $G = (V, E)$ be a connected graph.

Fact 1. If S is a 2-packing, then S is also a 2-packing of $G = (V, E \setminus e)$, for any edge e.

$\Rightarrow$ we can assume that G is a tree.

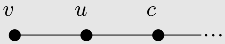
Fact 2. Suppose that there are two leaves which have a common neighbor. Every 2-packing in G is also a 2-packing in H.

$\Rightarrow$ we can assume that there are no two or more leaves with a common neighbor

... and Proper Pairs

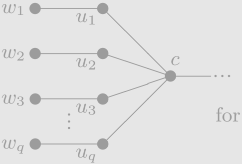
Proof. **2/2**

(A) If $\deg(c) \leq 2$ then

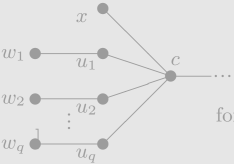


$$pp(n) \leq 2\, pp(n-1) + 4\, pp(n-3)$$

(B) If $\deg(c) > 2$ and



for $q \geq 2$



for $q \geq 1$

(B0) no neighbor of c is a leaf ...

$$pp(n) \leq 2^{2q}\, pp(n-2q) + (3^{q-1}2^{q+1}(3+q) - 2^{2q+1})\, pp(n-2q-1)$$

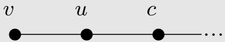
(B1) one neighbor of c is a leaf ...

$$pp(n) \leq 2^{2q+1}\, pp(n-2q-1) + (3^{q-1}2^{q+1}(9+2q) - 2^{2q+2})\, pp(n-2q-2)$$ □

... and Proper Pairs

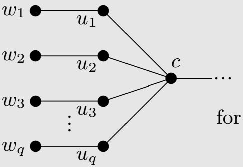
Proof. 2/2

(A) If $\deg(c) \leq 2$ then

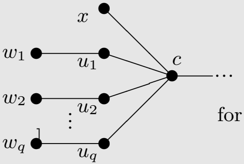


$$pp(n) \leq 2\,pp(n-1) + 4\,pp(n-3)$$

(B) If $\deg(c) > 2$ and



for $q \geq 2$



for $q \geq 1$

(B0) no neighbor of c is a leaf ...

$$pp(n) \leq 2^{2q}\,pp(n-2q) + (3^{q-1}2^{q+1}(3+q) - 2^{2q+1})\,pp(n-2q-1)$$

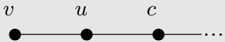
(B1) one neighbor of c is a leaf ...

$$pp(n) \leq 2^{2q+1}\,pp(n-2q-1) + (3^{q-1}2^{q+1}(9+2q) - 2^{2q+2})\,pp(n-2q-2) \quad \square$$

… and Proper Pairs

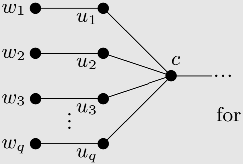
Proof. **2/2**

(A) If $\deg(c) \leq 2$ then

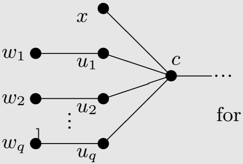


$$pp(n) \leq 2\,pp(n-1) + 4\,pp(n-3)$$

(B) If $\deg(c) > 2$ and



for $q \geq 2$



for $q \geq 1$

(B0) no neighbor of c is a leaf …

$$pp(n) \leq 2^{2q}\,pp(n-2q) + (3^{q-1}2^{q+1}(3+q) - 2^{2q+1})\,pp(n-2q-1)$$

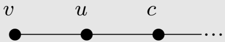
(B1) one neighbor of c is a leaf …

$$pp(n) \leq 2^{2q+1}\,pp(n-2q-1) + (3^{q-1}2^{q+1}(9+2q) - 2^{2q+2})\,pp(n-2q-2) \qquad \square$$

... and Proper Pairs

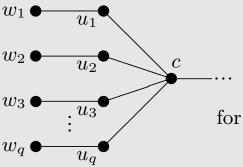
Proof. 2/2

(A) If $\deg(c) \leq 2$ then

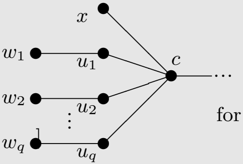


$$pp(n) \leq 2\, pp(n-1) + 4\, pp(n-3)$$

(B) If $\deg(c) > 2$ and



for $q \geq 2$



for $q \geq 1$

(B0) no neighbor of c is a leaf ...

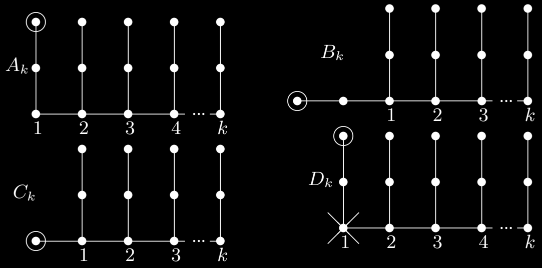
$$pp(n) \leq 2^{2q}\, pp(n-2q) + (3^{q-1}2^{q+1}(3+q) - 2^{2q+1})\, pp(n-2q-1)$$

(B1) one neighbor of c is a leaf ...

$$pp(n) \leq 2^{2q+1}\, pp(n-2q-1) + (3^{q-1}2^{q+1}(9+2q) - 2^{2q+2})\, pp(n-2q-2) \quad \square$$

… and Proper Pairs

To show the lower bound, we consider the following graphs :



$$\begin{cases} a_k = 2b_{k-1} + 4a_{k-1} \\ b_k = 2c_k + 2d_k \\ c_k = 2a_k + 12d_{k-1} \\ d_k = 4d_{k-1} + 12a_{k-1} \end{cases}$$

Theorem

$$2.6117^n \leq pp(n) \leq 2.6488^n$$

An Exact Exponential-Time Algorithm

One key ingredient of our algorithm

Main idea : Use algebraic manipulations similar to

fast matrix multiplication

Assume that A and B are $2^k \times 2^k$ matrices.

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \quad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \quad C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

where

$$\begin{aligned} C_{1,1} &= A_{1,1} \cdot B_{1,1} + A_{1,2} \cdot B_{2,1} \\ C_{1,2} &= A_{1,1} \cdot B_{1,2} + A_{1,2} \cdot B_{2,2} \\ C_{2,1} &= A_{2,1} \cdot B_{1,1} + A_{2,2} \cdot B_{2,1} \\ C_{2,2} &= A_{2,1} \cdot B_{1,2} + A_{2,2} \cdot B_{2,2} \end{aligned}$$

Thus, 8 matrix multiplications of $2^{k-1} \times 2^{k-1}$ matrices are necessary :

$$T(n) = 8 \cdot T(n/2) = O(n^3)$$

One key ingredient of our algorithm

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \quad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \quad C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

By Strassen [Stra69] :

$$\begin{aligned}
M_1 &= (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}) \\
M_2 &= (A_{2,1} + A_{2,2}) \cdot B_{1,1} \\
M_3 &= A_{1,1} \cdot (B_{1,2} - B_{2,2}) \\
M_4 &= A_{2,2} \cdot (B_{2,1} - B_{1,1}) \\
M_5 &= (A_{1,1} + A_{1,2}) \cdot B_{2,2} \\
M_6 &= (A_{2,1} - A_{1,1}) \cdot (B_{1,1} + B_{1,2}) \\
M_7 &= (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2})
\end{aligned}$$

and

$$\begin{aligned}
C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\
C_{1,2} &= M_3 + M_5 \\
C_{2,1} &= M_2 + M_4 \\
C_{2,2} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

Then,

$$T(n) = 7 \cdot T(n/2) = O(n^{2.807})$$

Our approach

Our algorithm uses Dynamic Programming

We reduce the number of operations (like in Strassen's algo)

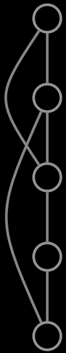
$+$

We use a representation for partial $L(2,1)$-labelings

Representation of partial $L(2,1)$-labelings

Span 1

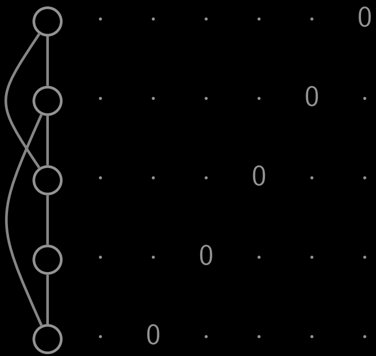
Table T_1



Representation of partial $L(2,1)$-labelings

Span 1

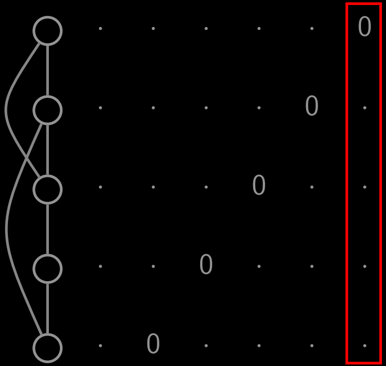
Table T_1



Representation of partial $L(2,1)$-labelings

Span 1

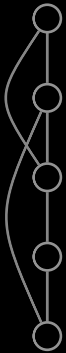
Table T_1



Representation of partial $L(2,1)$-labelings

Span 2

Table T_2

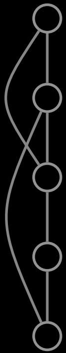


·	·	·	·	·	·	·	·	·	·	·	·	·	0	0	0	1	1	1
·	·	·	·	·	·	·	·	·	0	0	1	1	·	·	·	·	·	·
·	·	·	·	·	0	0	1	1	·	·	·	·	·	·	·	·	·	·
·	·	·	0	1	·	·	·	·	·	1	·	0	·	·	1	·	·	0
·	0	1	·	·	·	1	·	0	·	·	·	·	·	1	·	·	0	·

Representation of partial $L(2,1)$-labelings

Span 2

Table T_2

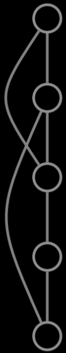


·	·	·	·	·	·	·	·	·	·	·	·	·	0	0	0	1	1	1
·	·	·	·	·	·	·	·	·	0	0	1	1	·	·	·	·	·	·
·	·	·	·	·	0	0	1	1	·	·	·	·	·	·	·	·	·	·
·	·	·	0	1	·	·	·	·	·	1	·	0	·	·	1	·	·	0
·	0	1	·	·	·	1	·	0	·	·	·	·	·	1	·	·	0	·

Representation of partial $L(2,1)$-labelings

Span 3

Table T_3



· ·0000000001111

· · · · · · · · · · · · · · · · · · · ·0000011122222 · · · · · · ·22 · · · ·

· · · · · · · · ·0000111 2222 · · · ·2 · · · · · · ·0 · · · · ·22 · · · · ·

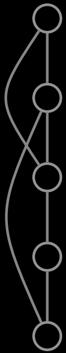
· · · ·00122 · · ·2 · · · · · ·0 · ·12 · ·02 · ·01 · · · ·12 · · ·1 · · ·0

·012 ·2 · ·0 ·12 · ·02 ·01 · ·2 · · · · · · · ·0 · · · ·12 · · ·1 · · ·02 ·

Representation of partial $L(2,1)$-labelings

Span 3

Table T_3



```
·································0000000001111
····················0000011122222·······22····
·········00001112222····2·······0·····22······
····00122···2······0··12··02··01····12···1···0
·012·2··0·12··02·01··2·······0····12···1···02·
```

Representation of partial $L(2,1)$-labelings

Jump to a compact representation

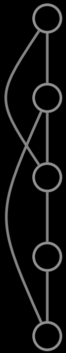
Table T_ℓ contains a vector $\vec{a} \in \{0, \overline{0}, 1, \overline{1}\}^n$ if and only if there is a partial labeling $\varphi\colon V \to \{0, \ldots, \ell\}$ such that :

- $a_i = 0$ iff v_i is not labeled by φ and there is no neighbor u of v_i with $\varphi(u) = \ell$
- $a_i = \overline{0}$ iff v_i is not labeled by φ and there is a neighbor u of v_i with $\varphi(u) = \ell$
- $a_i = 1$ iff $\varphi(v_i) < \ell$
- $a_i = \overline{1}$ iff $\varphi(v_i) = \ell$

Representation of partial $L(2,1)$-labelings

Span 3

Table T_3



$$000000000000000\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}11111111111\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}$$

$$00000000\bar{0}\bar{0}\bar{0}1111\bar{0}\bar{0}\bar{0}1\bar{1}\bar{1}\bar{1}\bar{1}00000\bar{0}\bar{0}\bar{0}\bar{0}\bar{1}\bar{1}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}11$$

$$000\bar{0}\bar{0}111001000\bar{0}\bar{1}\bar{1}\bar{1}\bar{1}\bar{0}\bar{0}\bar{0}1000\bar{0}\bar{0}00\bar{1}\bar{1}\bar{0}\bar{0}\bar{0}\bar{0}\bar{0}11\bar{0}\bar{0}$$

$$001\bar{1}\bar{1}00\bar{1}\bar{0}1\bar{0}0\bar{0}1\bar{1}\bar{0}\bar{0}1\bar{0}0010001\bar{1}\bar{1}\bar{0}1\bar{0}\bar{0}0100100 01$$

$$010\bar{0}101\bar{0}\bar{1}\bar{1}\bar{1}0\bar{1}0\bar{0}0100\bar{0}1\bar{0}\bar{0}010\bar{0}1\bar{1}\bar{1}01\bar{0}\bar{0}0100100$$

Computing the tables

How to compute table $T_{\ell+1}$ from table T_ℓ ?

Computing the tables

Let $P \subseteq \{0,1\}^n$ be the encodings of all 2-packings of G.

Formally, $\vec{p} \in P \Leftrightarrow \exists$ a 2-packing $S \subseteq V$ such that $\forall i$, $p_i = 1$ iff $v_i \in S$.

We compute $T_{\ell+1}$ from $T_\ell \oplus P$.

We define the partial function $\oplus\colon \{0,\bar{0},1,\bar{1}\} \times \{0,1\} \to \{0,1,\bar{1}\}$:

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	–	–

Entry "–" signifies that $\oplus$ is not defined.

We generalize $\oplus$ to vectors :

$$a_1 a_2 \ldots a_n \oplus b_1 b_2 \ldots b_n = \begin{cases} (a_1 \oplus b_1) \ldots (a_n \oplus b_n) & \text{if } \oplus \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Computing the tables

Then $T_\ell \oplus P$ is already almost the same as $T_{\ell+1}$:

$\vec{a} \in T_{\ell+1}$ iff there is an $\vec{a'} \in T_\ell \oplus P$ such that

- $a_i = 0$ iff $a'_i = 0$ and there is no $v_j \in N(v_i)$ with $a'_j = \overline{1}$
- $a_i = \overline{0}$ iff $a'_i = 0$ and there is a $v_j \in N(v_i)$ with $a'_j = \overline{1}$
- $a_i = 1$ iff $a'_i = 1$
- $a_i = \overline{1}$ iff $a'_i = \overline{1}$

Computing efficiently the tables

What remains is to find a method to compute $T_\ell \oplus P$

Computing efficiently the tables

What remains is to find a method to compute $T_\ell \oplus P$

fast

Computing efficiently the tables

Definition

$$A_w = \{\vec{v} \mid w \cdot v \in A\}$$

$\oplus$	0	$\overline{0}$	1	$\overline{1}$
0	0	0	1	1
1	$\overline{1}$	$\sim$	$-$	$-$

$$\begin{aligned} A \oplus B = \quad & 0((A_0 \cup A_{\overline{0}}) \oplus B_0) \\ \cup \quad & 1((A_1 \cup A_{\overline{1}}) \oplus B_0) \\ \cup \quad & \overline{1}(A_0 \oplus B_1) \end{aligned}$$

Computing efficiently the tables

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	$-$	$-$

for two adjacent vertices

for two adjacent vertices

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00																
01																
10																
11																

Computing efficiently the tables

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	$-$	$-$

for two adjacent vertices

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00																
01																
10																
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Computing efficiently the tables

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	–	–

for two adjacent vertices

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00																
01			–	–			–	–			–	–			–	–
10									–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Computing efficiently the tables

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	–	–

for two adjacent vertices

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00																
01		$\sim$	–	–		$\sim$	–	–		$\sim$	–	–		$\sim$	–	–
10					$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Computing efficiently the tables

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	–	–

for two adjacent vertices

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Computing efficiently the tables

$\oplus$	0	$\bar{0}$	1	$\bar{1}$
0	0	0	1	1
1	$\bar{1}$	$\sim$	$-$	$-$

for two adjacent vertices

for two adjacent vertices

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	$-$	$-$	$0\bar{1}$	$\sim$	$-$	$-$	$1\bar{1}$	$\sim$	$-$	$-$	$1\bar{1}$	$\sim$	$-$	$-$
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$\rightarrow$ Prefix $\bar{1}\bar{1}$ cannot appear.

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$A \oplus B =$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\overline{00}$	$\bar{0}1$	$\overline{01}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\overline{10}$	$\bar{1}1$	$\overline{11}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$A \oplus B = 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\overline{00}}) \oplus B_{00})$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	$-$	$-$	$0\bar{1}$	$\sim$	$-$	$-$	$1\bar{1}$	$\sim$	$-$	$-$	$1\bar{1}$	$\sim$	$-$	$-$
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$\begin{aligned} A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\bar{0}\bar{0}}) \oplus B_{00}) \\ \cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\bar{0}\bar{1}}) \oplus B_{00}) \end{aligned}$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\overline{00}$	$\bar{0}1$	$\overline{01}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\overline{10}$	$\bar{1}1$	$\overline{11}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$
\begin{aligned}
A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\overline{00}}) \oplus B_{00}) \\
\cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\overline{01}}) \oplus B_{00}) \\
\cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\overline{10}}) \oplus B_{00})
\end{aligned}
$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\overline{00}$	$\bar{0}1$	$\overline{01}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\overline{10}$	$\bar{1}1$	$\overline{11}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$
\begin{aligned}
A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\overline{00}}) \oplus B_{00}) \\
\cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\overline{01}}) \oplus B_{00}) \\
\cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\overline{10}}) \oplus B_{00}) \\
\cup \quad & 11((A_{11} \cup A_{1\bar{1}} \cup A_{\bar{1}1}) \oplus B_{00})
\end{aligned}
$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$\begin{aligned} A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\bar{0}\bar{0}}) \oplus B_{00}) \\ \cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\bar{0}\bar{1}}) \oplus B_{00}) \\ \cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\bar{1}\bar{0}}) \oplus B_{00}) \\ \cup \quad & 11((A_{11} \cup A_{1\bar{1}} \cup A_{\bar{1}1}) \oplus B_{00}) \\ \cup \quad & 0\bar{1}((A_{00} \cup A_{\bar{0}0}) \oplus B_{01}) \end{aligned}$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\overline{00}$	$\bar{0}1$	$\overline{01}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\overline{10}$	$\bar{1}1$	$\overline{11}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$
\begin{aligned}
A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\overline{00}}) \oplus B_{00}) \\
\cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\overline{01}}) \oplus B_{00}) \\
\cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\overline{10}}) \oplus B_{00}) \\
\cup \quad & 11((A_{11} \cup A_{1\bar{1}} \cup A_{\bar{1}1}) \oplus B_{00}) \\
\cup \quad & 0\bar{1}((A_{00} \cup A_{\bar{0}0}) \oplus B_{01}) \\
\cup \quad & 1\bar{1}((A_{10} \cup A_{\bar{1}0}) \oplus B_{01})
\end{aligned}
$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$
\begin{aligned}
A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\bar{0}\bar{0}}) \oplus B_{00}) \\
\cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\bar{0}\bar{1}}) \oplus B_{00}) \\
\cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\bar{1}\bar{0}}) \oplus B_{00}) \\
\cup \quad & 11((A_{11} \cup A_{1\bar{1}} \cup A_{\bar{1}1}) \oplus B_{00}) \\
\cup \quad & 0\bar{1}((A_{00} \cup A_{\bar{0}0}) \oplus B_{01}) \\
\cup \quad & 1\bar{1}((A_{10} \cup A_{\bar{1}0}) \oplus B_{01}) \\
\cup \quad & \bar{1}0((A_{00} \cup A_{0\bar{0}}) \oplus B_{10})
\end{aligned}
$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$
\begin{aligned}
A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\bar{0}\bar{0}}) \oplus B_{00}) \\
\cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\bar{0}\bar{1}}) \oplus B_{00}) \\
\cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\bar{1}\bar{0}}) \oplus B_{00}) \\
\cup \quad & 11((A_{11} \cup A_{1\bar{1}} \cup A_{\bar{1}1}) \oplus B_{00}) \\
\cup \quad & 0\bar{1}((A_{00} \cup A_{\bar{0}0}) \oplus B_{01}) \\
\cup \quad & 1\bar{1}((A_{10} \cup A_{\bar{1}0}) \oplus B_{01}) \\
\cup \quad & \bar{1}0((A_{00} \cup A_{0\bar{0}}) \oplus B_{10}) \\
\cup \quad & \bar{1}1((A_{01} \cup A_{0\bar{1}}) \oplus B_{10})
\end{aligned}
$$

Computing efficiently the tables

$\oplus$	00	$0\bar{0}$	01	$0\bar{1}$	$\bar{0}0$	$\bar{0}\bar{0}$	$\bar{0}1$	$\bar{0}\bar{1}$	10	$1\bar{0}$	11	$1\bar{1}$	$\bar{1}0$	$\bar{1}\bar{0}$	$\bar{1}1$	$\bar{1}\bar{1}$
00	00	00	01	01	00	00	01	01	10	10	11	11	10	10	11	-
01	$0\bar{1}$	$\sim$	–	–	$0\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–	$1\bar{1}$	$\sim$	–	–
10	$\bar{1}0$	$\bar{1}0$	$\bar{1}1$	$\bar{1}1$	$\sim$	$\sim$	$\sim$	$\sim$	–	–	–	–	–	–	–	–
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

$$
\begin{aligned}
A \oplus B = \quad & 00((A_{00} \cup A_{0\bar{0}} \cup A_{\bar{0}0} \cup A_{\bar{0}\bar{0}}) \oplus B_{00}) \\
\cup \quad & 01((A_{01} \cup A_{0\bar{1}} \cup A_{\bar{0}1} \cup A_{\bar{0}\bar{1}}) \oplus B_{00}) \\
\cup \quad & 10((A_{10} \cup A_{1\bar{0}} \cup A_{\bar{1}0} \cup A_{\bar{1}\bar{0}}) \oplus B_{00}) \\
\cup \quad & 11((A_{11} \cup A_{1\bar{1}} \cup A_{\bar{1}1}) \oplus B_{00}) \\
\cup \quad & 0\bar{1}((A_{00} \cup A_{\bar{0}0}) \oplus B_{01}) \\
\cup \quad & 1\bar{1}((A_{10} \cup A_{\bar{1}0}) \oplus B_{01}) \\
\cup \quad & \bar{1}0((A_{00} \cup A_{0\bar{0}}) \oplus B_{10}) \\
\cup \quad & \bar{1}1((A_{01} \cup A_{0\bar{1}}) \oplus B_{10})
\end{aligned}
$$

Running-time : $T(n) = 8 \cdot T(n-2) = 8^{n/2} < 2.8285^n$

Decomposing the graph into connected subgraphs

What about using a $\oplus$-table for $k' = O(1)$ vertices?

Imagine that a graph can be decomposed into some connected subsets of constant size k' ...

Decomposing the graph into connected subgraphs

Theorem ($\star$)

Let G be a connected graph of order n.
Let $k < n$ be a positive integer.

Then there exist connected subgraphs $G_1, G_2, \ldots, G_q$ of G s.t.

(i) every vertex of G belongs to at least one of them

(ii) the order of each of $G_1, G_2, \ldots, G_{q-1}$ is at least k and at most $2k$ (while for G_q we only require $|V(G_q)| \leq 2k$)

(iii) the sum of the numbers of vertices of $G_i's$ is at most $n(1 + \frac{1}{k})$

Decomposing the graph into connected subgraphs

Proof **1/2**

- Consider a DFS-tree T of G rooted at r.
- For every v let $T(v)$ be the subtree rooted in v.
- If $|T(r)| \leq 2k$ then add G to the set of desired subgraphs and stop.
- If there is a vertex v such that $k \leq |T(v)| \leq 2k$ then add $G[V(T(v))]$ to the set of desired subgraphs and proceed recursively with $G \setminus V(T(v))$.

Decomposing the graph into connected subgraphs

Proof **2/2**

- Otherwise there must be a vertex v such that $|T(v)| > 2k$ and for its every child u, $|T(u)| < k$.

  In such a case find a subset $\{u_1, \ldots, u_i\}$ of children of v such that $k-1 \leq |T(u_1)| + \cdots + |T(u_i)| \leq 2k-1$.

  Add $G[\{v\} \cup V(T(u_1)) \cup \cdots \cup V(T(u_i))]$ to the set of desired subgraphs and proceed recursively with $G \setminus (V(T(u_1)) \cup .. \cup V(T(u_i)))$.

- This procedure terminates after at most $\frac{n}{k}$ steps and in each of them we have left at most one vertex of the identified connected subgraph in the further processed graph.

$\square$

An exact algorithm

Let $A \subseteq \{0, \bar{0}, 1, \bar{1}\}^n$ and $B \subseteq \{0,1\}^n$ where $n > k'$.
We compute $A \oplus B$ is the following way :

$$A \oplus B = \bigcup_{\substack{\vec{u} \in \{0,\bar{0},1,\bar{1}\}^{k'} \\ \vec{v} \in \{0,1\}^{k'} \\ \text{s.t. } \vec{u} \oplus \vec{v} \text{ is defined}}} (\vec{u} \oplus \vec{v})(A_{\vec{u}} \oplus B_{\vec{v}})$$

$$= \bigcup_{\substack{\vec{v} \in \{0,1\}^{k'} \\ \vec{w} \in \{0,1,\bar{1}\}^{k'}}} \left[\left(\bigcup_{\substack{\vec{u} \in \{0,\bar{0},1,\bar{1}\}^{k'} \\ \text{s.t. } \vec{u} \oplus \vec{v} = \vec{w}}} A_{\vec{u}} \right) \oplus B_{\vec{v}} \right]$$

Remark :
Computation can be omitted whenever $\left(\bigcup_{\substack{\vec{u} \in \{0,\bar{0},1,\bar{1}\}^{k'} \\ \text{s.t. } \vec{u} \oplus \vec{v} = \vec{w}}} A_{\vec{u}} \right)$ is empty.

An exact algorithm – Running-time analysis

How many pairs $\vec{v}, \vec{w}$ are there s.t. there is at least one $\vec{u}$ with $\vec{u} \oplus \vec{v} = \vec{w}$?

If $\vec{v}$ is fixed, then $v_i = 1 \Rightarrow w_i = \overline{1}$.

Thus, for a fixed $\vec{v}$ there are at most $2^{k'-||\vec{v}||}$ many $\vec{w}$'s, where $||\vec{v}||$ denotes the number of positions i such that $v_i = 1$.

The total number of pairs $\vec{v}, \vec{w}$ such that $\vec{w} = \vec{v} \oplus \vec{u}$ for some $\vec{u}$ is therefore at most

$$\sum_{\vec{v} \in \{0,1\}^{k'}} 2^{k'-||\vec{v}||} \quad \leq \quad pp(k')$$

$\Rightarrow$ We need to make $pp(k')$ recursive computations of $\oplus$ on sets of vectors of length $n - k'$.

An exact algorithm – Running-time analysis

How many pairs $\vec{v}, \vec{w}$ are there s.t. there is at least one $\vec{u}$ with $\vec{u} \oplus \vec{v} = \vec{w}$?

If $\vec{v}$ is fixed, then $v_i = 1 \Rightarrow w_i = \overline{1}$.

Thus, for a fixed $\vec{v}$ there are at most $2^{k' - ||\vec{v}||}$ many $\vec{w}$'s, where $||\vec{v}||$ denotes the number of positions i such that $v_i = 1$.

The total number of pairs $\vec{v}, \vec{w}$ such that $\vec{w} = \vec{v} \oplus \vec{u}$ for some $\vec{u}$ is therefore at most

$$\sum_{\vec{v} \in \{0,1\}^{k'}} 2^{k' - ||\vec{v}||} \quad \leq \quad pp(k')$$

$\Rightarrow$ We need to make $pp(k')$ recursive computations of $\oplus$ on sets of vectors of length $n - k'$.

An exact algorithm – Running-time analysis

How many pairs $\vec{v}, \vec{w}$ are there s.t. there is at least one $\vec{u}$ with $\vec{u} \oplus \vec{v} = \vec{w}$?

If $\vec{v}$ is fixed, then $v_i = 1 \Rightarrow w_i = \overline{1}$.

Thus, for a fixed $\vec{v}$ there are at most $2^{k'-||\vec{v}||}$ many $\vec{w}$'s, where $||\vec{v}||$ denotes the number of positions i such that $v_i = 1$.

The total number of pairs $\vec{v}, \vec{w}$ such that $\vec{w} = \vec{v} \oplus \vec{u}$ for some $\vec{u}$ is therefore at most

$$\sum_{\vec{v} \in \{0,1\}^{k'}} 2^{k'-||\vec{v}||} \leq pp(k')$$

$\Rightarrow$ We need to make $pp(k')$ recursive computations of $\oplus$ on sets of vectors of length $n - k'$.

An exact algorithm – Running-time analysis

How many pairs $\vec{v}, \vec{w}$ are there s.t. there is at least one $\vec{u}$ with $\vec{u} \oplus \vec{v} = \vec{w}$?

If $\vec{v}$ is fixed, then $v_i = 1 \Rightarrow w_i = \overline{1}$.

Thus, for a fixed $\vec{v}$ there are at most $2^{k'-||\vec{v}||}$ many $\vec{w}$'s, where $||\vec{v}||$ denotes the number of positions i such that $v_i = 1$.

The total number of pairs $\vec{v}, \vec{w}$ such that $\vec{w} = \vec{v} \oplus \vec{u}$ for some $\vec{u}$ is therefore at most

$$\sum_{\vec{v} \in \{0,1\}^{k'}} 2^{k'-||\vec{v}||} \quad \leq \quad pp(k')$$

$\Rightarrow$ We need to make $pp(k')$ recursive computations of $\oplus$ on sets of vectors of length $n - k'$.

An exact algorithm – Running-time analysis

How many pairs $\vec{v}, \vec{w}$ are there s.t. there is at least one $\vec{u}$ with $\vec{u} \oplus \vec{v} = \vec{w}$?

If $\vec{v}$ is fixed, then $v_i = 1 \Rightarrow w_i = \overline{1}$.

Thus, for a fixed $\vec{v}$ there are at most $2^{k'-||\vec{v}||}$ many $\vec{w}$'s, where $||\vec{v}||$ denotes the number of positions i such that $v_i = 1$.

The total number of pairs $\vec{v}, \vec{w}$ such that $\vec{w} = \vec{v} \oplus \vec{u}$ for some $\vec{u}$ is therefore at most

$$\sum_{\vec{v} \in \{0,1\}^{k'}} 2^{k'-||\vec{v}||} \quad \leq \quad pp(k')$$

$\Rightarrow$ We need to make $pp(k')$ recursive computations of $\oplus$ on sets of vectors of length $n - k'$.

An exact algorithm – Running-time analysis

By Theorem $(\star)$, the total length of the vectors is $n' \leq n(1+1/k)$.

In each recursive computation :

- Prepare up to $pp(k')$ many pairs of sets of vectors of length $n' - k'$
- Recursively compute $\oplus$ on these pairs
- From the result, compute $T_{\ell+1}$ in linear time
- The size of B is at most $O(n2^{n'})$ bits
- The size of A is at most $O(npp(n'))$ bits :
  the $\overline{1}$'s form a 2-packing and there are only two possibilities (1 or $0/\overline{0}$) for the other nodes.

Thus the running-time is given by

$$T(n) \leq O(n \cdot pp(n') + pp(k') \cdot T(n' - k'))$$

where $k \leq k' \leq 2k$.

An exact algorithm – Running-time analysis

By Theorem ($\star$), the total length of the vectors is $n' \leq n(1+1/k)$.

In each recursive computation :

- Prepare up to $pp(k')$ many pairs of sets of vectors of length $n' - k'$
- Recursively compute $\oplus$ on these pairs
- From the result, compute $T_{\ell+1}$ in linear time
- The size of B is at most $O(n2^{n'})$ bits
- The size of A is at most $O(npp(n'))$ bits :
  the $\bar{1}$'s form a 2-packing and there are only two possibilities (1 or $0/\bar{0}$) for the other nodes.

Thus the running-time is given by

$$T(n) \leq O(n \cdot pp(n') + pp(k') \cdot T(n' - k'))$$

where $k \leq k' \leq 2k$.

An exact algorithm – Running-time analysis

The solution of

$$T(n) \leq O(n \cdot pp(n') + pp(k') \cdot T(n' - k'))$$

is

$$T(n) = O^*(pp(n')) = O^*(pp(n(1 + 1/k)))$$

Choosing constant k big enough :

$$T(n) = O(2.6488^n)$$

An exact algorithm – Running-time analysis

The solution of

$$T(n) \leq O(n \cdot pp(n') + pp(k') \cdot T(n' - k'))$$

is

$$T(n) = O^*(pp(n')) = O^*(pp(n(1 + 1/k)))$$

Choosing constant k big enough :

$$T(n) = O(2.6488^n)$$

Conclusion

Conclusion

- Combinatorial result : number of proper pairs

$$2.6117^n \leq pp(n) \leq 2.6488^n$$

- Exact exponential-time algorithm for $L(2,1)$-labelings

$$O(2.6488^n)$$

Interesting questions :

- Does inclusion/exclusion or subset convolution can achieve a $O(2^n)$-time algorithm ?
- Is it possible to find a 2-approx in $O(c^n)$ with $c \leq 2$?
- In [GY92], it is conjectured that $\lambda(G) \leq \Delta(G)^2$.
  It is still not fully resolved. It has been proved for graphs of large maximum degree [HRS08].

Merci !

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

UNIVERSITE D'ORLEANS

Bibliographie I

Bodlaender, H.L., Kloks, T., Tan, R.B., van Leeuwen, J. :
Approximations for lambda-Colorings of Graphs. Computer Journal 47 (2004), pp. 193–204.

Chang, G. J., Kuo, D. :
The $L(2,1)$-labeling problem on graphs.
SIAM Journal of Discrete Mathematics 9 (1996), pp. 309–316.

Cygan, M., Kowalik, L. :
Channel Assignment via Fast Zeta Transform.
arXiv :1103.2275

Fiala, J., Golovach, P., Kratochvíl, J. :
Distance Constrained Labelings of Graphs of Bounded Treewidth.
Proceedings of ICALP 2005, LNCS 3580 (2005), pp. 360–372.

Fiala, J., Kloks, T., Kratochvíl, J. :
Fixed-parameter complexity of λ-labelings.
Discrete Applied Mathematics 113 (2001), pp. 59–72.

Golovach, P., Kratsch, D., Couturier, J.-F. :
Coloring With Few Colors : Counting, Enumeration and Combinatorial Bounds.
Proceedings of WG 6410, LNCS 3580 (2010), pp. 39–50.

Griggs, J. R., Yeh, R. K. :
Labelling graphs with a condition at distance 2.
SIAM Journal of Discrete Mathematics 5 (1992), pp. 586–595.

Bibliographie II

Havet, F., Klazar, M., Kratochvíl, J., Kratsch, D., Liedloff, M. :
Exact Algorithms for $L(p, q)$-labelings of graphs.
submitted to STACS'09.

Havet, F., Klazar, M., Kratochvíl, J., Kratsch, D., Liedloff, M. :
Exact algorithms for $L(2, 1)$-labeling of graphs.
Algorithmica 59 (2011), pp. 169–194.

Havet, F., Reed, B., Sereni, J.-S. :
$L(2, 1)$-labellings of graphs.
Proceedings of SODA 2008 (2008), pp. 621–630.

Hell, P., Nešetřil, J. :
On the complexity of H-colouring,
Journal of Combinatorial Theory Series B 48 (1990), 92–110.

Janczewski, R., Kosowski, A., Małafiejski. M. :
The complexity of the $L(p, q)$-labeling problem for bipartite planar graphs of small degree.
Discrete Mathematics 309 (2009), pp. 3270–3279.

Junosza-Szaniawski, K., Kratochvíl, J., Liedloff, M., Rzążewski, P. :
Determining the $L(2, 1)$-Span in Polynomial Space.
proceedings of WG'12.

Kráľ, D. :
Channel assignment problem with variable weights.
SIAM Journal on Discrete Mathematics 20 (2006), pp. 690–704.

Bibliographie III

Roberts, F.S. :
private communication to J. Griggs.

Strassen, V. :
Gaussian Elimination is not Optimal.
Numerische Mathematik 13 (1969), pp. 354–356.