

Determining $L(2, 1)$ -Span in Polynomial Space

Frédéric Havet^a, Konstanty Junosza-Szaniawski^b, Martin Klazar^c, Jan Kratochvíl^c, Dieter Kratsch^d, Mathieu Liedloff^e, Paweł Rzażewski^{b,*}

^a*Projet Mascotte I3S (CNRS & UNSA) and INRIA, INRIA Sophia-Antipolis,
2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France*

^b*Warsaw University of Technology, Faculty of Mathematics and Information Science
Pl. Politechniki 1, 00-661 Warsaw, Poland*

^c*Department of Applied Mathematics, and Institute for Theoretical Computer Science,
Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic.*

^d*Laboratoire d'Informatique Théorique et Appliquée,
Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France*

^e*Laboratoire d'Informatique Fondamentale d'Orléans,
Université d'Orléans, 45067 Orléans Cedex 2, France.*

Abstract

An $L(2, 1)$ -labeling of a graph is a labeling of its vertices by nonnegative integers such that the labels of adjacent vertices differ by at least 2 and labels assigned to vertices with a common neighbor differ by at least 1. The span of a labeling is the difference of the smallest and largest labels used. It is known that finding an minimum span of any given graph is NP-Complete.

In this paper we present an algorithm for this problem, which works in time $O(7.9463^n)$ and uses only polynomial memory. This is the first exact algorithm for $L(2, 1)$ -labeling problem with time complexity $O(c^n)$ for some constant c and polynomial space complexity.

Keywords: $L(2, 1)$ -labeling, exact algorithm, polynomial space

1. Introduction

A frequency assignment problem is the problem of assigning channels of frequency (represented by nonnegative integers) to each radio transmitter, so that no transmitters interfere with each other. Hale [15] formulated this problem in terms of so-called T -coloring of graphs.

*Corresponding author

Email addresses: frederic.havet@sophia.inria.fr (Frédéric Havet),
k.szaniawski@mini.pw.edu.pl (Konstanty Junosza-Szaniawski),
klazar@kam.ms.mff.cuni.cz (Martin Klazar), honza@kam.ms.mff.cuni.cz (Jan Kratochvíl),
kratsch@univ-metx.fr (Dieter Kratsch), liedloff@univ-orleans.fr (Mathieu Liedloff),
p.rzazewski@mini.pw.edu.pl (Paweł Rzażewski)

According to [13], Roberts was the first who proposed a modification of this problem, which is called an $L(2, 1)$ -labeling problem. It asks for such a labeling with nonnegative integer labels, that no vertices in distance 2 in a graph have the same label and labels of adjacent vertices differ by at least 2.

A span of an $L(2, 1)$ -labeling is the difference between largest and smallest label used. By $\lambda(G)$ we denote an $L(2, 1)$ -span of G , which is the smallest span over all feasible $L(2, 1)$ -labelings of G .

The problem of $L(2, 1)$ -labeling has been extensively studied (see [5, 9, 12, 22] for some surveys on the problem and its generalizations). A considerable attention has been given to bounding the value of $\lambda(G)$ by some function of G .

Griggs and Yeh in their seminal paper [13] proved that $\lambda(G) \leq \Delta^2 + \Delta$ ¹ and conjectured, that $\lambda(G) \leq \Delta^2$ for every graph G . There are several results supporting this conjecture, for example Gonçalves [11] proved that $\lambda(G) \leq \Delta^2 + \Delta - 2$ for graphs with $\Delta \geq 3$. Havet *et al.* [16] have settled the conjecture in affirmative for graphs with $\Delta \geq 10^{69}$. For graphs with smaller Δ , the conjecture still remains open.

The second main branch of research in $L(2, 1)$ -labeling was pointed to analyzing the problem from the complexity point of view. Griggs and Yeh [13] proved that computing $\lambda(G)$ is an NP-hard problem. Fiala *et al.* [8] later proved that deciding $\lambda(G) \leq k$ remains NP-complete for every fixed $k \geq 4$ (for $k \leq 3$ the problem is polynomial). It remains NP-complete even for regular graphs (see Fiala and Kratochvíl [10]), planar graphs (see Eggeman *et al.* [6]) or series-parallel graphs (see Fiala *et al.* [7]).

An exact algorithm for the so called Channel Assignment Problem, presented by Král' [21], implies an $O^*(4^n)$ ² algorithm for the $L(2, 1)$ -labeling problem. Havet *et al.* [17] presented an algorithm for computing $\lambda(G)$, which works in time $O(3.8730^n)$. This algorithm has been improved [19, 20], achieving a complexity bound $O(3.2361^n)$. Recently, a new algorithm for $L(2, 1)$ -labeling with a complexity bound $O(2.6488^n)$ has been presented [18].

All algorithms mentioned above are based on dynamic programming approach and use exponential memory. Havet *et al.* [17] presented a branching

¹ Δ denotes the largest vertex degree in a graph

²In the O^* notation we omit polynomially bounded terms.

algorithm which determines if $\lambda(G) \leq k$ in time $O^*((k - 2.5)^n)$ and polynomial space. Until now, no algorithm for $L(2, 1)$ -labeling with time complexity $O(c^n)$ for some constant c and polynomial space complexity has been presented. However, there are such algorithms for a related problem of classical graph coloring. The first one, with time complexity $O(5.2830^n)$, was shown by Bodleander and Kratsch [4]. The best currently known algorithm for graph coloring with polynomial space complexity is by Björklund *et al.* [3], using the inclusion-exclusion principle. Its time complexity is $O(2.2461^n)$.

The main result of the following paper is the following theorem:

Theorem 1. *An $L(2, 1)$ -span of any graph on n vertices can be computed in time $O(7.9463^n)$ and polynomial space.*

The algorithm presented in this paper is the the first exact algorithm for $L(2, 1)$ -labeling problem with time complexity $O(c^n)$ for a constant c and polynomially bounded space complexity.

2. Notation

Throughout the paper we consider finite undirected graphs without multiple edges or loops. The symbol n is reserved for the number of vertices of the input graph, which will always be denoted by G . The vertex set (edge set) of a graph G is denoted by $V(G)$ ($E(G)$), respectively).

Let $N(v) = \{u \in V(G) : (u, v) \in E(G)\}$ denote the set of neighbors (the *neighborhood*) of a vertex v . The neighborhood of a set X of vertices in G is denoted by $N(X) = \bigcup_{v \in X} N(v)$. A subset $X \subseteq V(G)$ fulfilling $N(u) \cap (N(v) \cup \{v\}) = \emptyset$ for all $u, v \in X$, is called a *2-packing* in G . Note that in any $L(2, 1)$ -labeling of a graph G , the vertices with the same label form a 2-packing.

For a graph G , a *G -correct partition of a set $Y \subseteq V(G)$* is a triple (A, X, B) , such that:

1. The sets $A, X, B \subseteq Y$ form a partition of Y ,
2. X is a nonempty 2-packing in G ,
3. $|A| \leq \frac{|Y|}{2}$ and $|B| \leq \frac{|Y|}{2}$.

3. Algorithm

In this section we present a recursive algorithm for $L(2, 1)$ -labeling problem, which uses only polynomial space. To simplify the description, we will label the vertices with numbers starting from 1 (not from 0 as it is usually done).

In fact the algorithm presented in this section solves a somewhat more general problem described as follows. Suppose that the given instance is a subset of vertices $Y \subseteq V(G)$ of a graph G together with two subsets $Z, M \subseteq V(G)$. Let $k\text{-}\tilde{L}_Z^M(Y)$ -labeling of G denote a partial $L(2, 1)$ -labeling of Y with labels $\{1, \dots, k\}$, such that vertices of M obtain label at most $k - 1$ and vertices of Z obtain label at least 2. Let $\Lambda_Z^M(Y, G)$ denote the smallest possible k admitting the existence of such a labeling. We define $\Lambda_Z^M(\emptyset, G) \stackrel{\text{def.}}{=} 0$ for all graphs G and sets $Z, M \subseteq V(G)$. The generalized problem asks to compute an $\Lambda_Z^M(Y, G)$. Note that the problem does not care of the labels of $V(G) \setminus Y$.

Any $k\text{-}\tilde{L}_Z^M(Y)$ -labeling of G with $k = \Lambda_Z^M(Y, G)$ is called *optimal*.

We observe that even if c is an optimal $\tilde{L}_Z^M(Y)$ -labeling of G , then any of the sets $c^{-1}(1)$ and $c^{-1}(\Lambda_Z^M(Y, G))$ may be empty. In the extremal case, if $Z = M = Y$, then $c^{-1}(1) = c^{-1}(k) = \emptyset$ for all k and feasible $k\text{-}\tilde{L}_Z^M(Y)$ -labelings c of G . Moreover, notice that $\Lambda_\emptyset^\emptyset(G) = \lambda(G) + 1$.

The idea of the algorithm is to use the classical paradigm *Divide-and-Conquer* to design exponential-time algorithms. This paradigm was widely and successfully used to achieve many polynomial-time algorithms (e.g., the classical merge-sort algorithm). As shown in [1, 2, 14], by using the technique some NP-hard problems like Exact SAT, TSP or Graph Coloring can be solved in exponential-time needing only polynomial-space.

Lemma 1. *For every $k\text{-}\tilde{L}_Z^M(Y)$ -labeling c of G , there exists $\ell \in \{1, \dots, k\}$, such that the triple $(\bigcup_{i=1}^{\ell-1} c^{-1}(i), c^{-1}(\ell), \bigcup_{i=\ell+1}^k c^{-1}(i))$ is a G -correct partition of Y . Let us denote this value of ℓ by $\text{mid}(c)$.*

Proof. Notice that for every $\ell \in \{1, \dots, k\}$ the sets $\bigcup_{i=1}^{\ell-1} c^{-1}(i)$, $c^{-1}(\ell)$ and $\bigcup_{i=\ell+1}^k c^{-1}(i)$ clearly form a partition of Y and $c^{-1}(\ell)$ is a 2-packing. Let ℓ be the smallest number, such that $|\bigcup_{i=1}^{\ell} c^{-1}(i)| \geq \frac{|Y|}{2}$. Clearly $c^{-1}(\ell) \neq \emptyset$, because otherwise we would choose $\ell - 1$. Hence $|\bigcup_{i=0}^{\ell-1} c^{-1}(i)| \leq \frac{|Y|}{2}$. Moreover, the fact that $|\bigcup_{i=1}^{\ell} c^{-1}(i)| \geq \frac{|Y|}{2}$ implies that $|\bigcup_{i=\ell+1}^k c^{-1}(i)| \leq \frac{|Y|}{2}$. \square

First, the algorithm exhaustively checks if $\Lambda_Z^M(Y, G) \leq 3$. If not, the set Y is partitioned into three sets A, X, B , which form a G -correct partition of Y . The sets A and B are then labeled recursively.

The labeling of the whole Y is constructed from the labelings found in the recursive calls. The sets of labels used on the sets A and B are separated from each other by the label used for the 2-packing X . This allows to solve the subproblems for A and B independently from each other.

Iterating over all G -correct partitions of Y , the algorithm computes the minimum k admitting the existence of a k - $\tilde{L}_Z^M(Y)$ -labeling of G , which is by definition $\Lambda_Z^M(Y, G)$.

Algorithm 1: Find-Lambda

Input : Graph G , Sets $Y, Z, M \subseteq V(G)$

```

1 if  $Y = \emptyset$  then return 0
2 for  $k \leftarrow 1$  to 3 do
3   if there exists a  $k$ - $\tilde{L}_Z^M(Y)$ -labeling of  $G$  then return  $k$ 
4  $k \leftarrow \infty$ 
5 foreach  $G$ -correct partition  $(A, X, B)$  of  $Y$  do
6   if  $A \neq \emptyset$  and  $B \neq \emptyset$  then  $k_X \leftarrow 1$ 
7   if  $A = \emptyset$  and  $X \cap Z = \emptyset$  then  $k_X \leftarrow 1$ 
8   if  $A = \emptyset$  and  $X \cap Z \neq \emptyset$  then  $k_X \leftarrow 2$ 
9   if  $B = \emptyset$  and  $X \cap M = \emptyset$  then  $k_X \leftarrow 1$ 
10  if  $B = \emptyset$  and  $X \cap M \neq \emptyset$  then  $k_X \leftarrow 2$ 
11   $k_A \leftarrow \mathbf{Find-Lambda}(G, A, Z, N(X))$ 
12   $k_B \leftarrow \mathbf{Find-Lambda}(G, B, N(X), M)$ 
13   $k \leftarrow \min(k, k_A + k_X + k_B)$ 
14 return  $k$ 

```

Lemma 2. For a graph G and sets $Y, Z, M \subseteq V(G)$, if Y is a 2-packing in G , then $\Lambda_Z^M(Y, G) \leq 3$.

Proof. The labeling $c: Y \rightarrow \{1, 1, 3\}$ such that $c(v) = 2$ for every $v \in Y$ is a 3- $\tilde{L}_Z^M(Y)$ labeling of G . \square

Lemma 3. For any graph G and sets $Y, Z, M \subseteq V(G)$, the algorithm call $\mathbf{Find-Lambda}(G, Y, Z, M)$ returns $\Lambda_Z^M(Y, G)$.

Proof. If $Y = \emptyset$, the correct result is given in line 1 (by the definition of $\Lambda_Z^M(\emptyset, G)$). If $\Lambda_Z^M(Y, G) \leq 3$, the result is found by the exhaustive search in line 3. Notice that if $|Y| \leq 1$, then by Lemma 2 $\Lambda_Z^M(Y, G) \leq 3$.

Assume that the statement is true for all graphs G' and all sets $Y', Z', M' \subseteq V(G')$, such that $|Y'| < y$, where $y \geq 1$.

Let G be a graph and Y, Z, M be subsets of $V(G)$ such that $|Y| = y$. We may assume that $\Lambda_Z^M(Y, G) > 3$. Let k be the value returned by the algorithm call **Find-Lambda**(G, Y, Z, M).

First we prove that $k \geq \Lambda_Z^M(Y, G)$, i.e. there exists a k - $\tilde{L}_Z^M(Y)$ -labeling of G . Let us consider the G -correct partition (A, X, B) of Y , for which the value of k was set in line 13. Since each of the sets A and B has less than y vertices, by the inductive assumption there exists a k_A - $\tilde{L}_Z^{N(X)}(A)$ -labeling c_A of G and a k_B - $\tilde{L}_{N(X)}^M(B)$ -labeling c_B of G .

One of the following cases occurs:

1. If $A \neq \emptyset$ and $B \neq \emptyset$, then in line 6 the value of k_X is set to 1 and thus $k = k_A + k_B + 1$. The labeling c of Y , defined as follows:

$$c(v) = \begin{cases} c_A(v) & \text{if } v \in A \\ k_A & \text{if } v \in X \\ k_A + 1 + c_B(v) & \text{if } v \in B \end{cases}$$

is a k - $\tilde{L}_Z^M(Y)$ -labeling of G .

2. If $A = \emptyset$ and $X \cap Z = \emptyset$, then in line 7 the value of k_X is set to 1 and thus $k = k_B + 1$. The labeling c of Y , defined as follows:

$$c(v) = \begin{cases} 1 & \text{if } v \in X \\ c_B(v) + 1 & \text{if } v \in B \end{cases}$$

is a k - $\tilde{L}_Z^M(Y)$ -labeling of G .

3. If $A = \emptyset$ and $X \cap Z \neq \emptyset$, then in line 8 the value of k_X is set to 2 and thus $k = k_B + 2$. The labeling c of Y , defined as follows:

$$c(v) = \begin{cases} 2 & \text{if } v \in X \\ c_B(v) + 2 & \text{if } v \in B \end{cases}$$

is a k - $\tilde{L}_Z^M(Y)$ -labeling of G .

4. If $B = \emptyset$ and $X \cap M = \emptyset$, then in line 9 the value of k_X is set to 1 and thus $k = k_A + 1$. The labeling c of Y , defined as follows:

$$c(v) = \begin{cases} c_A(v) & \text{if } v \in A \\ k_A + 1 & \text{if } v \in X \end{cases}$$

is a $k\text{-}\tilde{L}_Z^M(Y)$ -labeling of G .

5. If $B = \emptyset$ and $X \cap M \neq \emptyset$, then in line 10 the value of k_X is set to 2 and thus $k = k_A + 2$. The labeling c of Y , defined as follows:

$$c(v) = \begin{cases} c_A(v) & \text{if } v \in A \\ k_A + 1 & \text{if } v \in X \end{cases}$$

is a $k\text{-}\tilde{L}_Z^M(Y)$ -labeling of G (the label $k_A + 2$ is counted as used, but no vertex is labeled with it).

The case when $X = \emptyset$ is not possible, since the partition (A, X, B) is G -correct. The case when $A = B = \emptyset$ is not possible, since then $Y = X$ is a 2-packing in G and by Lemma 2 $\Lambda_Z^N(Y, G) \leq 3$, so the algorithm would finish in line 3.

Now let us show that $k \leq \Lambda_Z^M(Y, G)$. Let c be an optimal $\tilde{L}_Z^M(Y)$ -labeling of G and $\ell = \text{mid}(c)$. Let $A = c^{-1}(1) \cup \dots \cup c^{-1}(\ell - 1)$, $X = c^{-1}(\ell)$ and $B = c^{-1}(\ell + 1) \cup \dots \cup c^{-1}(\Lambda_Z^M(Y, G))$. Since the triple (A, X, B) is a G -correct partition of Y by Lemma 1, the algorithm considers it in one of the iterations of the main loop.

Let $c_A: A \rightarrow \mathbb{N}$ be a function such that $c_A(v) = c(v)$ for every $v \in A$ and $c_B: B \rightarrow \mathbb{N}$ be a function such that $c_B(v) = c(v) - \ell$ for every $v \in B$. Notice that c_A is an optimal $\tilde{L}_Z^{N(X)}(A)$ -labeling of G and c_B is an optimal $\tilde{L}_{N(X)}^M(B)$ -labeling of G , because otherwise c would not be an optimal.

Hence by the inductive assumption the call in line 11 returns the number $k_A \leq \Lambda_Z^{N(X)}(A, G)$ and the call in line 12 returns the number $k_B \leq \Lambda_{N(X)}^M(B, G)$.

Let k' be the value of $k_A + k_X + k_B$ in the iteration of the main loop when partition (A, X, B) is considered.

Let us consider the following cases:

1. $A, B \neq \emptyset$. In such a case the algorithm **Find-Lambda** sets $k_X = 1$ in line 6 and

$$\Lambda_Z^M(Y, G) = \Lambda_Z^{N(X)}(A, G) + \underbrace{1}_{c^{-1}(\ell)=X} + \Lambda_Z^{N(X)}(B, G) \geq k_A + k_X + k_B = k'.$$

2. $A = \emptyset$ and $\ell = 1$. In such a case $k_A = 0$ and $X \cap Z = \emptyset$ and the algorithm

Find-Lambda sets $k_X = 1$ in line 7 and

$$\Lambda_Z^M(Y, G) = \underbrace{\Lambda_Z^{N(X)}(A, G)}_{=0} + \underbrace{1}_{c^{-1}(1)=X} + \Lambda_Z^{N(X)}(B, G) \geq k_A + k_X + k_B = k'.$$

3. $A = \emptyset$ and $\ell = 2$. In such a case $k_A = 0$ and $X \cap Z \neq \emptyset$. Otherwise c' defined by $c'(v) = c(v) - 1$ for every $v \in Y$ would be an $\tilde{L}_Z^M(Y)$ -labeling of G using less labels than the optimal $\tilde{L}_Z^M(Y)$ -labeling c of G – contradiction. The algorithm **Find-Lambda** sets $k_X = 2$ in line 8 and

$$\Lambda_Z^M(Y, G) = \underbrace{\Lambda_Z^{N(X)}(A, G)}_{=0} + \underbrace{1}_{c^{-1}(1)=\emptyset} + \underbrace{1}_{c^{-1}(2)=X} + \Lambda_Z^{N(X)}(B, G) \geq k_A + k_X + k_B = k'.$$

4. $B = \emptyset$ and $\ell = \Lambda_Z^M(Y, G)$. In such a case $k_B = 0$ and $X \cap M = \emptyset$, and the algorithm **Find-Lambda** sets $k_X = 1$ in line 9 and

$$\Lambda_Z^M(Y, G) = \Lambda_Z^{N(X)}(A, G) + \underbrace{1}_{c^{-1}(\Lambda_M^N(Y, G))=X} + \underbrace{\Lambda_Z^{N(X)}(B, G)}_{=0} \geq k_A + k_X + k_B = k'.$$

5. $B = \emptyset$ and $\ell = \Lambda_Z^M(Y, G) - 1$. In such a case $k_B = 0$ and $X \cap M \neq \emptyset$ and the algorithm **Find-Lambda** sets $k_X = 2$ in line 10 and

$$\Lambda_Z^M(Y, G) = \Lambda_Z^{N(X)}(A, G) + \underbrace{1}_{c^{-1}(\Lambda_M^N(Y, G)-1)=X} + \underbrace{1}_{c^{-1}(\Lambda_M^N(Y, G))=\emptyset} + \underbrace{\Lambda_Z^{N(X)}(B, G)}_{=0} \geq k_A + k_X + k_B = k'.$$

Since those are all possible cases and k is the minimum over values of k' for all G -correct partitions of Y , clearly $k \leq \Lambda_Z^M(Y, G)$. \square

Now let us estimate the computational complexity of our algorithm.

Lemma 4. *The algorithm **Find-Lambda** computes $\Lambda_Z^M(Y, G)$ of any graph in time $O((9 + \epsilon)^n)$ and polynomial space for all $\epsilon > 0$.*

Proof. Verifying if a given $X \subseteq Y$ is a 2-packing in G can be performed in polynomial time. We can check if a given function $c: Y \rightarrow \mathbb{N}$ is an $\tilde{L}_Z^M(Y)$ -labeling of G in polynomial time as well.

Let $y = |Y|$ be the measure of the size of the problem. Let $T(y)$ denote the running time of the algorithm **Find-Lambda** applied to a graph G and $Y, Z, M \subseteq V(G)$. The algorithm **Find-Lambda** first checks in constant time if $Y = \emptyset$. Then it exhaustively checks if there exists a k - $\tilde{L}_Z^M(Y)$ -labeling of G for $k \in \{1, 2, 3\}$. There are 3^y functions $c: Y \rightarrow \{1, 2, 3\}$, so this step is performed

in time $O(y^{O(1)} \cdot 3^y)$. Then for every G -correct partition of Y the algorithm is called recursively for two sets of size at most $y/2$. Notice that there are at most 3^y considered partitions. Checking if a partition of Y is G -correct can be performed in polynomial time. Hence we obtain the following inequality for the complexity:

$$T(y) \leq 3^y y^{O(1)} T(y/2) \quad (1)$$

Since $y \leq n$, the solution of this recurrence is bounded by $O(9^n n^{O(1) \log n}) = O(9^n 2^{O(1) \log^2 n})$ which is bounded by $O((9 + \epsilon)^n)$, for all $\epsilon > 0$. The space complexity of the algorithm is clearly polynomial. \square

We can refine the complexity analysis to prove Theorem 1.

Notice that if G is disconnected, we can label each of its components separately. Hence we can assume that G is connected in the first call of the algorithm (however, it may become disconnected in recursive calls).

In [18] a notion of *proper pairs* has been introduced. A pair of disjoint subsets (P, Q) of $V(G)$ is *proper* if P is a 2-packing. It has been shown in [18] that the maximum number of proper pairs in a connected graph on n vertices, denoted by $pp(n)$, is bounded by $O(2.6488^n)$. Notice that each proper pair (P, Q) corresponds bijectively to a G -correct partition $(Q, P, V(G) \setminus (P \cup Q))$ of $V(G)$. Hence the number of G -correct partitions of $Y \subseteq V(G)$ for a connected graph G is bounded by $pp(n) = O(2.6488^n)$.

Hence the complexity of the algorithm is bounded by:

$$T'(n) \leq 3^n n^{O(1)} + 2.6488^n \left(3^n n^{O(1)} T(n/2) \right), \quad (2)$$

where T is given by inequality (1). From (2) we obtain $T'(n) = O(7.9463^n)$.

References

- [1] Björklund, A., and Husfeldt, T., Exact Algorithms for Exact Satisfiability and Number of Perfect Matchings. *Proceedings of ICALP 2006*, LNCS 4051, Springer, 2006, pp. 548–559.
- [2] Björklund, A., and Husfeldt, T., Exact Algorithms for Exact Satisfiability and Number of Perfect Matchings. *Algorithmica* 52 (2008), pp. 226–249.

- [3] Björklund, A., Husfeldt, T., Koivisto, M.: Set Partitioning via Inclusion-Exclusion. *SIAM J. Comput.* 39 (2009), pp. 546–563
- [4] Bodlaender, H.L., Kratsch D.: An exact algorithm for graph coloring with polynomial memory. *UU-CS 2006-015* (2006)
- [5] Calamoneri, T.: The $L(h, k)$ -Labelling Problem: An Updated Survey and Annotated Bibliography. *Computer Journal* (2011), doi:10.1093/comjnl/bxr037
- [6] Eggeman, N., Havet, F., Noble, S.: k - $L(2, 1)$ -Labelling for Planar Graphs is NP-Complete for $k \geq 4$. *Discrete Applied Mathematics* 158 (2010), pp. 1777–1788
- [7] Fiala, J., Golovach, P., Kratochvíl, J.: Distance Constrained Labelings of Graphs of Bounded Treewidth. *Proceedings of ICALP 2005*, LNCS 3580 (2005), pp. 360–372
- [8] Fiala, J., Kloks, T., Kratochvíl, J.: Fixed-parameter complexity of λ -labelings. *Discrete Applied Mathematics* 113 (2001), pp. 59–72
- [9] Fiala, J., Kratochvíl, J.: Locally constrained graph homomorphisms - structure, complexity, and applications. *Computer Science Review* 2 (2008), pp. 97–111
- [10] Fiala, J., Kratochvíl, J.: On the Computational Complexity of the $L(2, 1)$ -Labeling Problem for Regular Graphs *Proceedings of ICTCS 2005*, LNCS 3703 (2005), pp. 228–236
- [11] Gonçalves, D.: On the $L(p; 1)$ -labelling of graphs. *Discrete Mathematics* 308 (2008), pp. 1405–1414
- [12] Griggs, J. R., Král, D.: Graph labellings with variable weights, a survey. *Discrete Applied Mathematics* 157 (2009), pp. 2646–2658
- [13] Griggs, J. R., Yeh, R. K.: Labelling graphs with a condition at distance 2. *SIAM Journal of Discrete Mathematics* 5 (1992), pp. 586–595
- [14] Gurevich, Y., and Shelah, S.: Expected computation time for Hamiltonian path problem. *SIAM Journal on Computing* 16 (1987), pp. 486–502.

- [15] Hale, W.K.: Frequency assignment: Theory and applications. *Proc. IEEE* 68 (1980), pp. 1497–1514
- [16] Havet, F., Reed, B., Sereni, J.-S.: $L(2, 1)$ -labellings of graphs. *Proceedings of SODA 2008* (2008), pp. 621–630
- [17] 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
- [18] Junosza-Szaniawski K., Kratochvíl J., Liedloff M., Rossmanith P., Rzażewski P.: Fast Exact Algorithm for $L(2, 1)$ -Labeling of Graphs. *Proceedings of TAMC 2011*, LNCS 6648 (2011), pp. 82–93
- [19] Junosza-Szaniawski K., Rzażewski P.: On Improved Exact Algorithms for $L(2, 1)$ -Labeling of Graphs. *Proceedings of IWOCA 2010*, LNCS 6460 (2011), pp. 34–37
- [20] Junosza-Szaniawski K., Rzażewski P.: On the Complexity of Exact Algorithm for $L(2, 1)$ -labeling of Graphs. *Information Processing Letters* 111 (2011), pp. 697–701
- [21] D. Král': Channel assignment problem with variable weights. *SIAM Journal on Discrete Mathematics* 20 (2006), pp. 690–704
- [22] Yeh, R.: A survey on labeling graphs with a condition at distance two. *Discrete Mathematics* 306 (2006), pp. 1217–1231