An Exact Algorithm for the Channel Assignment Problem

Daniel Kráľ[*]

Abstract

A channel assignment problem is determined by a triple (V, E, w) where V is a vertex set, E an edge set and w is a function assigning edges positive integer weights. An assignment c of integers between 1 and K to the vertices is proper if $|c(u) - c(v)| \geq w(uv)$ for each $uv \in E$; the smallest K for which there is a proper assignment is the span of the problem. The problem is l-bounded if the values of w do not exceed l.

An algorithm running in time $O(n(l+2)^n)$ for computing the span of an l-bounded channel assignment problem with n vertices is presented and we discuss (im)possibility of designing a faster algorithm based on maximal independent set approach. We further find an algorithm running in time $O(nl(l+2)^n)$ for computing the number of different proper assignments of span at most K.

1 Introduction

The channel assignment problem naturally generalizes graph coloring. A *channel assignment problem* is a triple (V, E, w) where V is a *vertex set*, E is a set consisting of pairs of elements of V (called *edges*) and w is a function assigning edges of E positive integer weights. If e is an edge of E, the number $w(e)$ is called the *weight* of the edge e. The value of $w(uv)$ is considered to be zero if $uv \notin E$. An *assignment* c of positive integers to the vertices, i.e. the elements of V, is *proper*, if $|c(u) - c(v)| \geq w(uv)$

[*]Department of Applied Mathematics and Institute for Theoretical Computer Science (Institute for Theoretical Computer Science is supported by Ministry of Education of Czech Republic as project LN00A056), Charles University, Malostranské náměstí 25, 118 00 Prague, Czech Republic. E-mail: `kral@kam.mff.cuni.cz`.

for each edge $uv \in E$. The *span of a proper assignment* is the maximum integer assigned to a vertex; the *span of a channel assignment problem* is the smallest integer k for which there is a proper assignment of the span k. If all the edge weights are equal to one, then an assignment is proper iff it is a coloring of the underlying graph $G = (V, E)$ and the span of this channel assignment problem is the chromatic number of G. A channel assignment problem and its special case called distance graph labelling [6, 7] can be used to model real-world problems: The vertices can be thought as radio transmitters, integers assigned to them as frequencies and the weights of the edges as minimal differences of frequencies which can be assigned to the transmitters corresponding to their end-vertices without interference.

Determining the span of a channel assignment problem is very hard; it is NP-complete even if the underlying graph has treewidth 3 (cf. [12]). A channel assignment problem is *l-bounded* if the weight of each edge is at most l. In Section 3, we find an algorithm for computing an optimal assignment of an l-bounded channel assignment running in time $O(n(l+2)^n)$ and in space $O((l+2)^n)$ where $n = |V|$ (Corollary 1). This improves a previous algorithm due to McDiarmid running in time $O(n^2(2l+1)^n)$ from [11], see also [10]. Our algorithm is based on a concept of separated sequences introduced in Section 2. Next, we modify the algorithm to an algorithm running in time $O(nl(l+2)^n)$ which computes the number of proper assignments of span at most K (Theorem 2); note that the running time of the counting algorithm is independent on K. The computational models considered in this paper are the following: In case of the first algorithm, we assume that we are capable to store n-bit pointers and each operation with $O(\log n)$-bit numbers requires a unit of time. In case of the second algorithm, since the resulting numbers might be $\omega(\log n)$-bit numbers, we assume that each operation with $\Theta(b)$-bit numbers requires a unit of time, if the results are $\Theta(b)$-bit numbers. The just presented models are implicitly assumed throughout the paper.

For the case of ordinary graph coloring (a 1–bounded channel assignment problem), our algorithm runs in time $O(3^n)$ (omitting the polynomial factor). There are better algorithms for this special case: The first one with running time $O(2.4422^n)$ was obtained by Lawler in [8] and an improvement to $O(2.4150^n)$ from [5] is due to Eppstein. All these algorithms are based on maximal independent set (MIS) technique. There is a series of papers on algorithms for 3–coloring of graphs [1, 2, 4, 13] and only one of them, [1], is not MIS–based. A question might be whether it is possible to improve the running time $O(n(l+2)^n)$ of our algorithm using the MIS technique.

We show that this is impossible in Section 4: There are 2–bounded channel assignment problems such that any MIS–based algorithm outputs a span arbitrarily larger than the actual span is (Theorem 3).

2 Separated Sequences

In this section, definitions and notation related to separated sequences are introduced.

Let (V, E, w) be a fixed l-bounded channel assignment in this paragraph. We define $w(uv) := 0$ in case that $uv \notin E$. A sequence $\mathcal{V} = V_0, \ldots, V_{k-1}$ of k subsets of V is *non-decreasing* if $V_{i-1} \subseteq V_i$ for $1 \leq i \leq k-1$. A sequence $\mathcal{V}$ is *separated* if it is non-decreasing and $w(v_i v_j) \leq j - i$ for any $v_i \in V_i \setminus V_{i-1}$, $v_j \in V_j \setminus V_{j-1}$, $1 \leq i \leq j \leq k-1$. In particular, if $\mathcal{V}$ is separated, then $V_i \setminus V_{i-1}$ is an independent set for any $1 \leq i \leq k-1$. Note that there is no condition on the weights of the edges joining the vertices of V_0 to the remaining vertices. Unless stated otherwise, a (non-decreasing, separated) sequence is a sequence formed by subsets of the vertex set in this paper. A subsequence of a separated sequence is separated, too. A non-decreasing sequence $\mathcal{V}' = V'_0, \ldots, V'_{k'-1}$ *extends* a non-decreasing sequence $\mathcal{V} = V_0, \ldots, V_{k-1}$ if there exists $i_0, k - k' + 1 \leq i_0 \leq k-1$ such that the following holds:

$$V_{i_0+j} = V'_j \text{ for } 0 \leq j \leq k - 1 - i_0.$$

The sequence $\mathcal{V}'' = V_0, \ldots, V_{i_0} = V'_0, \ldots, V_{k-1} = V'_{k-1-i_0}, \ldots, V'_{k'-1}$ is called an *extension* of $\mathcal{V}$ with $\mathcal{V}'$ and write $\mathcal{V} \oplus \mathcal{V}'$ for it; it will be always clear from the context which extension we have in mind if there are more possible extensions. The extension of a separated sequence with another separated sequence is not necessarily a separated sequence in general. However, the following holds:

Lemma 1 *Let (V, E, w) be an l-bounded channel assignment, $\mathcal{V}'$ a separated sequence of length at least $l + 1$ which extends a separated sequence $\mathcal{V}$. If the length $\mathcal{V} \oplus \mathcal{V}'$ is the length of $\mathcal{V}$ increased by one, then $\mathcal{V} \oplus \mathcal{V}'$ is separated.*

Proof: Let $\mathcal{V} \oplus \mathcal{V}' = V_0, \ldots, V_K$, $1 \leq i, j \leq K$, $v_i \in V_i \setminus V_{i-1}$ and $v_j \in V_j \setminus V_{j-1}$. If both $i < K$ and $j < K$, then $w(v_i v_j) \leq |i - j|$ because $\mathcal{V}$ is separated. Let further $i = K$. If $j \leq K - l$, then $w(v_i v_j) \leq l \leq |i - j|$

because the problem is l-bounded. If $j > K - l$, then $w(v_i v_j) \leq |i-j|$ because $\mathcal{V}'$ is separated.

■

The span of a channel assignment problem is related to separated sequences in the next lemma:

Lemma 2 *Let (V, E, w) be a channel assignment problem and let $K \geq 0$ be an integer. Proper assignments of (V, E, w) with the span at most K one-to-one correspond to separated sequences $V_0, \ldots, V_K$ of length $K+1$ with $V_0 = \emptyset$ and $V_K = V$.*

Proof: Let c be a proper assignment. Let $V_i = \{v | c(v) \leq i\}$ for $0 \leq i \leq k$. The just defined sequence is separated since $V_i \setminus V_{i-1} = \{v | c(v) = i\}$ and $|c(u) - c(v)| \geq w(uv)$ for all the edges uv of E; the conditions $V_0 = \emptyset$ and $V_K = V$ are also satisfied. On the other hand, we can define for a separated sequence $V_0 = \emptyset, V_1, \ldots, V_{K-1}, V_K = V$ a proper assignment $c(v) = \min\{i | v \in V_i\}$. Since $V_0 = \emptyset$ and $V_K = V$, the assignment c is defined for all the vertices and its span is at most K. Because the sequence is separated, c is proper. It is straightforward to verify that the just defined correspondence is really one-to-one.

■

The immediate corollary of Lemma 2 is the following:

Lemma 3 *The span of a channel assignment problem (V, E, w) is equal to the minimum K such that there exists a separated sequence $V_0, \ldots, V_K$ with $V_0 = \emptyset$ and $V_K = V$.*

3 The algorithms

We describe our algorithms for l-bounded channel assignment problems in this section. Let (V, E, w) be a given l-bounded channel assignment problem. We write $\mathcal{G}_l(V, E, w)$ for the directed graph whose vertices are all the separated sequences of length l and a sequence $\mathcal{V}$ is joined by an arc to a sequence $\mathcal{V}'$ if $\mathcal{V}'$ extends $\mathcal{V}$ and $\mathcal{V} \oplus \mathcal{V}'$ is a separated sequence of length $l+1$. Let $\mathcal{V}_l^{\emptyset}$ be the (separated) sequence $\emptyset, \ldots, \emptyset$ of length l and $\mathcal{V}_l^V$ be the (separated) sequence $V, \ldots, V$ of length l. Note that $\mathcal{G}_l(V, E, w)$ might contain loops, in particular, there are loops incident with $\mathcal{V}_l^{\emptyset}$ and $\mathcal{V}_l^V$. $\mathcal{V}_l^{\emptyset}$

is called the *initial separated sequence* and $\mathcal{V}_l^V$ the *final separated sequence*. We first show that $\mathcal{G}_l(V,E,w)$ can be constructed in time $O(|V|(l+2)^{|V|})$:

Lemma 4 *Let (V,E,w) be a channel assignment problem. There is an algorithm which outputs all the separated sequences of length k and which runs in time $O(n(k+1)^n)$ and in space $O(n)$ where $n=|V|$. The number of separated sequences of length k is at most $(k+1)^n$.*

Proof: The algorithm is in Figure 1; it uses the following one-to-one correspondence between sequences $a_1,\dots,a_n$ of integers between 0 and k of length n and non-decreasing sequences $V_0,\dots,V_{k-1}$ of length k:

$$a_i = \min\{j|v_i \in V_j\} \cup \{k\}$$

The algorithm simply generates all such sequences of integers and checks that the corresponding sequence $\mathcal{V}$ of sets is separated. One can check that the edges v_iv_j, $1 \le j < i$, do not violate the condition that $\mathcal{V}$ is separated, when generating a_i. This straightforwardly leads to our $O(n(k+1)^n)$-time algorithm.

■

Lemma 5 *Let (V,E,w) be an l-bounded channel assignment problem and $n := |V|$. The graph $\mathcal{G}_l(V,E,w)$ can be constructed in time $O(n(l+2)^n)$ and in space $O((l+2)^n)$. The graph $\mathcal{G}_l(V,E,w)$ contains at most $(l+2)^n$ arcs.*

Proof: We create a complete $(l+1)$-ary tree $\mathcal{T}$ of depth n (the depth of a tree containing only a root is zero). The paths from the root to the leaves in $\mathcal{T}$ correspond to sequences of integers between 0 and l and hence to non-decreasing sequences of subsets of V of length l as described in the proof of Lemma 4. Notice that it is possible to find for a sequence $\mathcal{V}$ the corresponding leaf of $\mathcal{T}$ in time $O(n)$. We use twice the algorithm from Lemma 4. First, all the separated sequences of length l are generated and stored in the leaves of $\mathcal{T}$ whether the corresponding sequence is separated. Next, there will be stored a list of arcs leading from the corresponding vertex of $\mathcal{G}_l(V,E,w)$ in each leaf of $\mathcal{T}$: All the separated sequences $\mathcal{V}$ of length $l+1$ are generated and split to two sequences $\mathcal{V}'$ and $\mathcal{V}''$ of length l each such that $\mathcal{V}=\mathcal{V}'\oplus\mathcal{V}''$ ($\mathcal{V}$ can be uniquely decomposed to $\mathcal{V}'$ and $\mathcal{V}''$) and we add to the list of arcs leading from $\mathcal{V}'$ stored in $\mathcal{T}$ an arc from $\mathcal{V}'$ to $\mathcal{V}''$. Finally, the graph

```
Input:          channel assignment problem (V,E,w), n:=|V|
                integer k
Initial call:   gener(1)

var a: array[1..n] of integer;

procedure gener(next: integer);
  var i,j: integer;
  if next=n+1 then
    for i:=1 to n do
      if a[i]<k then
        output "the i-th vertex is in all sets V_j j>=a[i]"
      fi
    endfor
    exit
  else
    for i:=0 to k do
      a[next]:=i
      for j:=1 to next-1 do
        if a[j]<>0 and a[j]<>k and a[next]<>0 and a[next]<>k and
           |a[next]-a[j]|<w(v_{next}v_j) then
          continue the for-cycle for next i
        fi
      endfor
      gener(next+1)
    endfor
    exit
```

Figure 1: The algorithm for generating separated sequences of length k

$\mathcal{G}_l(V,E,w)$ is output by a simple single traversing $\mathcal{T}$. The running time of the algorithm is dominated by the step when the algorithm from Lemma 4 is used for the second time and it is $O(n(l+2)^n)$.

Since each arc of $\mathcal{G}_l(V,E,w)$ corresponds to a separated sequence of length $l+1$, the number of arcs of $\mathcal{G}_l(V,E,w)$ is at most $(l+2)^n$ due to Lemma 4. From this, it is clear that space $O((l+2)^n)$ is enough to store $\mathcal{G}_l(V,E,w)$ and hence it covers all space demands of the algorithm.

■

We are now ready to relate $\mathcal{G}_l(V,E,w)$ to proper assignments:

Theorem 1 *Let (V,E,w) be an l-bounded channel assignment problem. Then the length of the shortest directed path between $\mathcal{V}_l^{\emptyset}$ and $\mathcal{V}_l^{V}$ in $\mathcal{G}_l(V,E,w)$ is equal to the span of (V,E,w) increased by $l-1$.*

Proof: Let $\mathcal{V}_0,\ldots,\mathcal{V}_L$ be a shortest directed path from $\mathcal{V}_l^{\emptyset}$ to $\mathcal{V}_l^{V}$ in $\mathcal{G}_l(V,E,w)$. It follows from the definition of $\mathcal{G}_l(V,E,w)$ that $\mathcal{V}_{i-1}\oplus\mathcal{V}_i$ is a separated sequence of length $l+1$ $(1\le i\le L)$. Since (V,E,w) is l-bounded, it straightforwardly follows from Lemma 1 that $\mathcal{V}_0\oplus\ldots\oplus\mathcal{V}_L$ is a separated sequence of length $L+l$. We can construct a separated sequence $\mathcal{V}$ of length $L+l-2(l-1)$ by cutting first $l-1$ and last $l-1$ elements of $\mathcal{V}_0\oplus\ldots\oplus\mathcal{V}_L$. Since the first element of $\mathcal{V}$ is $\emptyset$ and the last one is V, the span of (V,E,w) is at most $L-l+1$ due to Lemma 3.

We prove the opposite inequality of the statement of the theorem in this paragraph. Let K be the span of (V,E,w) and $\mathcal{V}$ a separated sequence of length $K+1$ starting with $\emptyset$ and ending with V (it exists due to Lemma 3). $\mathcal{V}$ is extended to $\mathcal{V}'$ by adding $l-1$ empty sets to the beginning of $\mathcal{V}$ and by adding $l-1$ sets V to the end. The length of $\mathcal{V}'$ is $K+2l-1$. $\mathcal{V}'$ can be split to sequences $\mathcal{V}_0,\ldots,\mathcal{V}_L$ of length l formed by consecutive subsequences of $\mathcal{V}'$. Note that $\mathcal{V}_0=\mathcal{V}_l^{\emptyset}$, $\mathcal{V}_L=\mathcal{V}_l^{V}$ and $L=K+l-1$. This establishes the theorem.

■

Theorem 1 implies together with Lemma 2 and Lemma 5 the following:

Corollary 1 *There is an algorithm for computing the span of an l-bounded channel assignment problem and constructing an optimal assignment which runs in time $O(n(l+2)^n)$ and space $O((l+2)^n)$ where n is the number of vertices.*

Proof: The proof of Theorem 1 actually describes the way how to find an optimal assignment of the problem. The shortest path between two vertices in a directed graph can be found in time linear in the number of arcs, see, e.g., [3].

■

The first algorithm can be modified to count proper assignments:

Theorem 2 *There is an algorithm for computing the number of proper assignments of span at most K for a given l-bounded channel assignment problem and given K; the algorithm runs in time $O(nl(l+2)^n)$ and in space $O((l+2)^n)$ where n is the number of vertices of the problem.*

Proof: Proper assignments of span at most K one-to-one correspond to separated sequences of length $K+1$ starting with an empty set and ending with the whole vertex set due to Lemma 2. It follows from the proof of Theorem 1 that separated sequences of length $K+1$ one-to-one correspond to directed walks (i.e., they may contain the same vertex more times) between $\mathcal{V}_l^{\emptyset}$ and $\mathcal{V}_l^{V}$ of length $K+l-1$ in $\mathcal{G}_l(V,E,w)$. We reduced the problem to computing the number of directed walks in $\mathcal{G}_l(V,E,w)$.

First, the vertices of $\mathcal{G}_l(V,E,w)$ are topologically sorted, i.e., a sequence of vertices $\mathcal{V}^1,\ldots,\mathcal{V}^N$ of $\mathcal{G}_l(V,E,w)$ is found which satisfies that if $\mathcal{V}^i\mathcal{V}^j$ is an arc, then $i \le j$. This can be done in time $O((l+2)^n)$, see, e.g., [3]. Note that $\mathcal{V}^1 = \mathcal{V}_l^{\emptyset}$ and $\mathcal{V}^N = \mathcal{V}_l^{V}$. For each vertex $\mathcal{G}_l(V,E,w)$, a triple (A,p,k_0) is computed where A is an array of size k_0 indexed by numbers from 0 to k_0-1, p is a polynomial and k_0 is an integer. We write (A^i,p^i,k_0^i) for a triple assigned to the vertex $\mathcal{V}^i$. It is demanded that the triple (A^i,p^i,k_0^i) assigned to $\mathcal{V}^i$ satisfies the following:

$$\begin{array}{l}\text{The number of directed walks from the vertex}\\ \mathcal{V}^1 = \mathcal{V}_l^{\emptyset} \text{ to the vertex } \mathcal{V}^i \text{of length } k \text{ is}\end{array} \begin{cases} A^i[k] & \text{if } 0 \le k < k_0. \\ p^i(k) & \text{if } k_0 \le k. \end{cases}$$

The triples can be computed in the order determined by the topological ordering. The triple assigned to $\mathcal{V}^1$ consists of an empty array A^1, a polynomial $p^1(k) := 1$ and an integer $k_0 := 0$. Let us assume that the triples for all $\mathcal{V}^i$, $i < i_0$ have been computed. We first compute a triple (A',p',k_0') which has the same meaning as $(A^{i_0},p^{i_0},k^{i_0})$ except that it counts the directed walks from $\mathcal{V}^1$ to $\mathcal{V}^{i_0}$ which end with exactly one $\mathcal{V}^{i_0}$. Let I be the set of those i for which there is an arc $\mathcal{V}^i\mathcal{V}^{i_0}$ in $\mathcal{G}_l(V,E,w)$. The following holds due to the definition of the triples:

$$
\begin{aligned}
k'_0 &:= 1 + \max_{i \in I} k^i \\
A'[0] &:= 0 \\
A'[k] &:= \sum_{i \in I, k-1 < k_0^i} A^i[k-1] + \sum_{i \in I, k-1 \geq k_0^i} p^i(k-1) \text{ for } 1 \leq k < k'_0 \\
p'(k) &:= \sum_{i \in I} p^i(k-1)
\end{aligned}
$$

If there is not a loop $\mathcal{V}^{i_0}\mathcal{V}^{i_0}$ in $\mathcal{G}_l(V,E,w)$, then any walk from $\mathcal{V}^1$ to $\mathcal{V}^{i_0}$ ends with exactly one $\mathcal{V}^{i_0}$ and hence $(A^{i_0}, p^{i_0}, k^{i_0}) = (A', p', k'_0)$. If there is a loop $\mathcal{V}^{i_0}\mathcal{V}^{i_0}$, then the triple $(A^{i_0}, p^{i_0}, k^{i_0})$ is determined as follows:

$$
\begin{aligned}
k_0^{i_0} &:= k'_0 \\
A^{i_0}[k] &:= \sum_{0 \leq k' \leq k} A'[k'] \text{ for } 0 \leq k < k_0 \\
p^{i_0}(k) &:= \sum_{0 \leq k' < k'_0} A'[k'] + \sum_{k'_0 \leq k' \leq k} p'(k')
\end{aligned}
$$

Using the above formulas, we can compute all the triples (note that there are explicit formulas for summing polynomials of bounded degree which can be used in case of computing $p^{i_0}(k)$) and we can determine the number of directed walks from $\mathcal{V}^1 = \mathcal{V}_l^{\emptyset}$ to $\mathcal{V}^N = \mathcal{V}_l^V$ from (A^N, p^N, k_0^N).

It remains to establish the time and space demands of the algorithm. We first realize that the length of any directed path in $\mathcal{G}_l(V,E,w)$ is at most nl: For any path $\mathcal{V}_1, \ldots, \mathcal{V}_k$ the sequence $\mathcal{V}_1 \oplus \ldots \oplus \mathcal{V}_k$ is a separated (and hence non-decreasing) sequence of length $k+l-1$ and it does not contain a subsequence consisting of at least $l+1$ identical sets (otherwise the sequence $\mathcal{V}_1, \ldots, \mathcal{V}_k$ contained the same vertex twice). This immediately gives that $k+l-1 \leq nl$ and hence $k \leq nl$. Since k_0^i is the length of the longest directed path from $\mathcal{V}^1$ to $\mathcal{V}^i$ (this can be easily proved by induction from the formulae defining k_0^i for $1 \leq i \leq N$), the above implies that $k_0^i \leq nl$ for all $1 \leq i \leq N$. The degree of a polynomial is increased only at a vertex with a loop; hence the degree of p^i is at most the maximum number of vertices with loops on a path from $\mathcal{V}^1$ to $\mathcal{V}^i$. On any directed path $\mathcal{V}_1, \ldots, \mathcal{V}_k$ there are at most n vertices with loops because the vertices $\mathcal{V}^i$ with loops are precisely the vertices corresponding to sequences consisting of l identical subsets of V, and the sequence $\mathcal{V}_1 \oplus \ldots \oplus \mathcal{V}_k$ is separated (and hence non-decreasing). We may conclude that the degree of any p^i is at most n. The time and space needed by the algorithm can be now estimated: The graph $\mathcal{G}_l(V, E, w)$ can be constructed in time $O(n(l+2)^n)$ and in space $O((l+2)^n)$. Computation of (A', p', k') requires addition of at most nl values for each arc leading to a vertex and hence it takes total time $O(nl(l+2)^n)$ for all the vertices of $\mathcal{G}_l(V, E, w)$ because $\mathcal{G}_l(V, E, w)$ contains at most $(l+2)^n$ arcs. In addition at every vertex with a loop, we need extra time to compute the actual triple from (A', p', k'). This extra time is at most $O(l \cdot p(n))$ for some polynomial $p(n)$. Since $\mathcal{G}_l(V, E, w)$ contains at most $(l+1)^n$ vertices, the time bound $O(nl(l+2)^n)$ dominates the extra time $O(l(l+1)^n p(n))$. The space needed by the algorithm consists of major parts: the space needed to store $\mathcal{G}_l(V, E, w)$ which is $O((l+2)^n)$ and the space needed to store all the triples which is $O(nl(l+1)^n)$. The former expression dominates the latter.

■

Note that we actually proved that the number of proper satisfying assignments of span K is a polynomial in K from sufficiently large K. This result has been previously proved using other techniques in [9] and in [14].

4 MIS approach

One of successful approaches in designing faster algorithms for graph coloring is based on maximal (inclusionwise) independent sets (MIS), see [2, 4, 5, 8, 13]. A general scheme of such a coloring algorithm can be found in Figure 2. The tricky part of the algorithms from [2, 4, 5, 8, 13] is elimination of certain small configurations in a given graph and then bounding the number of maximal independent sets (and hence the running time of the algorithm). A general scheme of an MIS-based algorithm for a channel assignment problem can be found in Figure 3. We prove that the span computed by an MIS-based algorithm can be arbitrarily larger than the actual span of a given problem even if restricted to 2-bounded channel assignment problems.

We first introduce several definitions: If $G_1 = (V_1, E_1, w_1)$ and $G_2 = (V_2, E_2, w_2)$ are channel assignment problems, then a k-join of G_1 and G_2 is the following channel assignment problem:

$$V = V_1 \cup V_2$$

```
Initial call:  chromatic_number(G, 0, empty set)

function chromatic_number(graph G, integer i,
                          precolored vertices W): integer;
  if W=V(G)
    return i
  fi
  k:=infinity
  for all maximal independent sets A of vertices in G-W do
    color vertices of A with the color i+1
    k':=chromatic_number(G, i+1, W united with A)
    if k'<k then
      k:=k'
    fi
  endfor
  return k
```

Figure 2: A scheme of an MIS-based algorithm for graph coloring

$$E = E_1 \cup E_2 \cup \{v_1 v_2 | v_1 \in V_1, v_2 \in V_2\}$$
$$w(e) = w_i(e_i) \text{ if } e_i \in E_i \text{ and } w(e) = k\text{otherwise}$$

We simply add all possible edges between the vertex sets V_1 and V_2 and assign them the weights equal to k. Let $G_1 = (V_1, E_1, w_1)$, $G_4 = (V_4, E_4, w_4)$ and $G_7 = (V_7, E_7, w_7)$ be the three channel assignment problems determined by following equalities (cf. Figure 4):

$$V_1 = \{a\} \text{ and } E_1 = \emptyset$$

$$V_4 = \{a, b, \alpha, \beta\}$$
$$E_4 = \{ab, \alpha\beta, a\alpha, a\beta, b\alpha, b\beta\}$$
$$w_4(a\alpha) = w_4(a\beta) = w_4(b\alpha) = w_4(b\beta) = 1$$
$$w_4(ab) = w_4(\alpha\beta) = 2$$
$$V_7 = \{a, b, c, \alpha, \beta, \alpha', \beta'\}$$
$$E_7 = \{ab, ac, bc, \alpha\beta, a\alpha, a\beta, b\alpha, b\beta, c\alpha, c\beta, \alpha\alpha', \beta\beta'\}$$
$$w_7(ab) = w_7(ac) = w_7(bc) = w_7(\alpha\beta) = w_7(\alpha\alpha') = w_7(\beta\beta') = 2$$

```
Initial call:  span(P, 0, empty assignment)

function span(channel assignment problem P, integer i,
              preassignment c): integer;
  if all the vertices have been assigned colors then
    return i
  fi
  k:=infinity
  for all maximal sets A of vertices which can be assigned in P
          the color i+1 with respect to the preassignment c do
    extend c to c' by assigning vertices of A the color i+1
    k':=span(P, i+1, c')
    if k'<k then
      k:=k'
    fi
  done
  return k
```

Figure 3: A scheme of an MIS-based algorithm for channel assignment problem

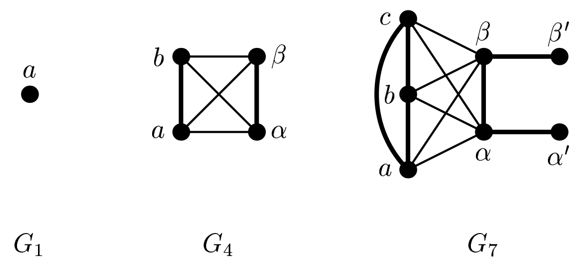

Figure 4: The channel assignment problems G_1, G_4 and G_7. The edges with the weights equal to 2 are drawn bold.

$$w_7(a\alpha) = w_7(a\beta) = w_7(b\alpha) = w_7(b\beta) = w_7(c\alpha) = w_7(c\beta) = 1$$

Let $G_{1,4,7}(A, B, C)$ is a channel assignment obtained by the mutual 2-join of A disjoint copies of G_1, B disjoint copies of G_4 and C disjoint copies of G_7. We keep the names of the vertices of the three just introduced channel assignment problems throughout the whole section.

Lemma 6 *The span of the channel assignment problem* $G_{1,4,7}(0, 0, C)$ *is at most* $6C - 1$.

Proof: There is a proper assignment of span 5 for the channel assignment problem G_7: Assign $1, 3, 5$ to the vertices a, b, c (respectively), the number 2 to both the vertices α and β' and the number 4 to both the vertices β and α'. It is easy to find (based on this assignment) a proper assignment of $G_{1,4,7}(0, 0, C)$ of span at most $6C-1$: Use the numbers $1, \ldots, 5$ for the first copy of G_7, the number $7, \ldots, 11$ for the second copy of G_7, the number $13, \ldots, 17$ for the third copy of G_7, etc.

■

Lemma 7 *An MIS-based algorithm outputs as the span of the channel assignment problem* $G_{1,4,7}(A, B, C)$ *a number which is at least* $2A+5B+7C-1$.

Proof: The proof proceeds by induction on the number of vertices of $G_{1,4,7}$. The claim is clear for $G_{1,4,7}(1, 0, 0)$ or $G_{1,5,7}(0, 1, 0)$. The remaining cases are reduced to smaller cases in the following. Consider what possible types of maximal (inclusionwise) independent sets of vertices in $G_{1,4,7}(A, B, C)$ are:

- **The set contains (and hence it consists of) the only vertex of a copy of G_1.**
  This vertex receives color 1, no vertex receives color 2 and the problem left is $G_{1,4,7}(A-1,B,C)$. The claim follows now straightforwardly from the induction.

- **The set contains of a vertex of a copy of G_4.**
  Let a, b, α and β be the vertices of the copy of G_4. Due to the symmetry, one may assume that the set contains (only) the vertex a. Hence a is assigned the color 1. In the next step, either α or β is assigned 2. Next, b is assigned 3 and in the last step the remaining of the vertices α and β is assigned 4. Then no vertex receives 5 and the problem which remains is $G_{1,4,7}(A,B-1,C)$. The algorithm determines as the span of the new problem a number which is at least $2A+5B+7C-6$ and as the span of the original problem $G_{1,4,7}(A,B,C)$ a number which is at least $2A+5B+7C-1$ as desired.

- **The set contains only vertices of a copy of G_7. Assume it contains either α or β.**
  It is possible to assume (w.l.o.g.) that the independent set is $\{\alpha,\beta'\}$. These two vertices receive the number 1. In the next step, one of the vertices a, b and c (say a) receives the number 2. Next, the vertices β and α' receive 3. In the next step, one of the vertices b or c (say b) receives 4 and then no vertex receives 5. We are left with a single vertex of the original copy of G_7, but it might happen that the algorithm proceeds with coloring vertices of another copy. Hence the induction has to be used for $G_{1,4,7}(A+1,B,C-1)$ to get the claim.

- **The set contains only vertices of a copy of G_7. Assume it contains one of the vertices a, b and c.**
  It is possible to assume (w.l.o.g.) that the vertex contained in the set is a. Then, the set is formed by a, α' and β'. These three vertices receive the colour 1 and in the next step none of the remaining vertices receivess the color 2. The problem which we are left with is $G_{1,4,7}(A,B+1,C-1)$ and the induction gives the claim.

- **The set contains only vertices of a copy of G_7. Assume it contains one of the vertices α', β'.**
  In such a case, the set also contains one of the vertices a, b, c, α and β and one of the previous cases applies.

■

Theorem 3 *For each $k \geq 0$, there is a 2-bounded channel assignment problem such that the span output by an MIS-based algorithm exceeds its actual span by at least k.*

Proof: It is enough to use Lemma 6 and Lemma 7 for $G_{1,4,7}(0,0,C)$ for C sufficiently large.

■

Acknowledgement

The author would like to thank Jan Kára and Jan Kratochvíl for careful reading a preliminary version of this manuscript.

References

[1] R. Beigel, D. Eppstein: *3-coloring in time $O(1.3446^n)$: a no-MIS algorithm*, Proc. 36th IEEE Symp. Foundations of Computer Science (FOCS), 1995, 444–453. ECCC TR95-033, `ftp://ftp.eccc.uni-trier.de/`.

[2] R. Beigel, D. Eppstein: *3-coloring in time $O(1.3289^n)$*. ACM Computing Research Repository, `cs.DS/0006046`, 2000.

[3] G. Chartrand, O. R. Oellermann: *Applied and Algorithmic Graph Theory*, McGraw-Hill, 1993.

[4] D. Eppstein: *Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction*, Proc. 12th Symp. Discrete Algorithms (SODA) 2001, 329–337.

[5] D. Eppstein: *Small Maximum Independent Sets and Faster Exact Graph Coloring*, Proc. 7th Workshop Algorithms and Data Structures, LNCS 2125, Springer, 2001, 462–470.

[6] J. Fiala, T.Kloks, J. Kratochvíl: *Fixed-parameter complexity of λ-labelings*, Discrete Applied Mathematics 113, 2001, 59–72.

[7] J. R. Griggs, R. K. Yeh: *Labelling graphs with a condition at distance 3*, SIAM J. Discrete Math, 5, 1992, 586–595.

[8] E. L. Lawler: *A note on the complexity of the chromatic number problem*, Inf. Proc. Lett. 5 (3), 1976, 66–67.

[9] C. McDiarmid: *Counting and constraint matrices*, manuscript, 1998.

[10] C. McDiarmid: *Discrete mathematics and radio channel assignment*, in: Recent advances in theoretical and applied discrete mathematics, Eds. C. Linhares-Salas and B. Reed, Springer, 2001.

[11] C. McDiarmid: *On the span in channel assignment problems: bounds, computing and counting*, submitted.

[12] C. McDiarmid, B. A. Reed: *Channel assignment and bounded treewidth graphs*, submitted.

[13] I. Schiermeyer: *Deciding 3-colourability in less than* $O(1.415^n)$ *steps*, Proc. 19th Int. Workshop on Graph-Theoretic Concepts in Computer Science, LNCS 790, Springer, 1994, 177–182.

[14] D. J. A. Welsh, G. Whittle: *Arrangements, channel assignments and associated polynomials*, Adv. Appl. Math. 23, 1999, 275–406.