Uniwersytet Jagielloński
Wydział Matematyki i Informatyki
INSTYTUT INFORMATYKI
Studia dzienne

Nr indeksu: WMIF/SMP/11/01/02

Grzegorz Gutowski

# On-line coloring of $k$-connected subgraphs

Opiekun pracy magisterskiej:
dr Maciej Ślusarek

Kraków 2006

# Streszczenie

Przydział częstotliwości radiowych w sieciach bezprzewodowych jest bardzo istotnym problemem praktycznym. Znając układ sieci możemy przedstawić ją w języku teorii grafów. Nadajnikom przypisujemy wierzchołki, które łączymy krawędziami gdy odpowiadające im nadajniki mogą nawiązać bezpośrednią łączność. Przydział częstotliwości nadajnikom musi zapewniać brak interferencji pomiędzy nimi i w prosty sposób tłumaczy się na klasyczny problem kolorowania grafu.

W klasycznym podejściu do przydziału częstotliwości, każde dwa *sąsiadujące* (mogące się ze sobą bezpośrednio komunikować) nadajniki muszą otrzymać różne częstotliwości. W tej pracy analizujemy pokrewny problem, w którym pozwalamy na przydział tej samej częstotliwości sąsiadującym wierzchołkom. W konsekwencji takiego przydziału tracą one jednak możliwość bezpośredniej komunikacji. Poszukiwany algorytm przydziału ma, używając jak najmniejszej liczby różnych częstotliwości, zachować odporność na uszkodzenia sieci (możliwe są zarówno uszkodzenia nadajników, jak i połączeń między nimi). Te wymagania w naturalny sposób tłumaczą się na terminy związane ze spójnością grafu.

Po wprowadzeniu niezbędnego aparatu matematycznego, przedstawiamy algorytm zachowujący $k$-spójność sieci przy użyciu $k+1$ kolorów dla $k < 3$. W głównej części pracy analizujemy problem, w którym do sieci są dynamicznie dodawane nowe nadajniki, a wcześniej przydzielone częstotliwości nie mogą już zostać zmienione. Mamy wtedy do czynienia z problemem *on-line*. Podajemy warunki, które muszą być zachowywane przez każdy algorytm poprawnie rozwiązujący taki problem.

Następnie opisujemy zachowanie zachłannego algorytmu *First-Fit* na grafach spełniających różne dodatkowe ograniczenia. Dowodzimy, że używa on co najwyżej $k+1$ kolorów do zachowania $k$-spójności, jeżeli sieć przez cały czas działania pozostaje $k$-spójna. Pokazujemy również, że dla grafów o średnicy ograniczonej przez $m$ algorytm First-Fit używa co najwyżej $\lfloor \frac{m+1}{2} \rfloor$ kolorów dla zachowania spójności.

# Introduction

Frequency allocation in the wireless communications networks is an important task in real-life applications. Given description of the network we can represent it in the language of graph theory. Transmitters are represented as vertices. Each pair of vertices that are able to communicate directly with each other is connected by an edge. Frequency allocation needs to assure that the transmitters do not interfere with each other. This demand is easily translated to the classical problem of finding a graph coloring.

The classical approach to the frequency allocation is to give different frequencies to *adjacent* (able to communicate directly) transmitters. We focus on a similar problem where the same frequency can be assigned to two adjacent transmitters. As a consequence of such allocation, the transmitters are no longer able to communicate directly with each other. We search for an algorithm minimizing the number of frequencies used which preserves the network fault tolerance (both transmitter and connection failures are possible). This extra requirement is captured by the notion of graph connectivity.

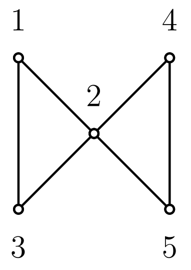
After introducing some basic graph theoretic notions, we present an algorithm preserving $k$-connectivity using $k+1$ colors for $k < 3$. In the main part we focus on the *on-line* version of the problem where new transmitters can be dynamically added to the network but once assigned frequencies cannot be changed. We study the possible scenarios of network growth and we are able to give strict rules that need to be followed by any algorithm solving the problem.

Later on we examine the behavior of the *First-Fit* algorithm in several cases where constructed networks satisfy some extra requirements. We prove that the First-Fit algorithm uses at most $k+1$ colors if the network is $k$-connected at every moment. We also prove that, if the network diameter is limited to $m$, then the First-Fit algorithm uses at most $\lfloor \frac{m+1}{2} \rfloor$ colors to preserve connectivity.

# 1 Basic notions and facts

A *graph* (undirected, loopless) is a pair $G = (V, E)$, where $V$ is a set and $E$ is a set of 2-element subsets of $V$. The elements of $V$ are *vertices* and the elements of $E$ are *edges*. The usual way to picture a graph is by drawing a dot for each vertex and joining two of these dots with a line if the corresponding two vertices form an edge. $|G|$ denotes the number of vertices.

Figure 1: Sample graph



Graph $G = (V, E)$ with $V = \{1, 2, 3, 4, 5\}$, and
$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{4, 5\}\}$

A vertex $v$ is *incident* to an edge $e$ if $v \in e$. The two vertices incident with an edge are its *ends*, and an edge *joins* its ends. An edge $\{x, y\}$ is usually written as $xy$ (or $yx$). Two vertices $x, y$ of $G$ are *adjacent*, or *neighbors*, if $xy$ is an edge of $G$. The *degree* $d(v)$ of a vertex $v$ is the number of neighbors of $v$. A vertex of degree 0 is *isolated*. The number $\delta(G) = min\{d(v) : v \in V\}$ is the *minimum degree* of $G$. If all vertices of $G$ are pairwise adjacent then $G$ is *complete*. A complete graph with $n$ vertices is denoted by $K_n$.

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. We call $G$ and $G'$ *isomorphic*, and write $G \simeq G'$, if there exists a bijection $f : V \to V'$ such that $\forall_{x,y \in V} xy \in E \Leftrightarrow f(x)f(y) \in E'$. We say that $G'$ is a *subgraph* of $G$ and write $G' \subseteq G$, when $E' \subseteq E$ and $V' \subseteq V$. If $G' \subseteq G$ and $G'$ contains all edges $xy \in E$ with $x, y \in V'$, then $G'$ is an *induced subgraph* of $G$, and $G[V'] := G'$. If $G' \subseteq G$ and $V' = V$ then $G'$ is a *spanning* subgraph of $G$. If $U$ is a set of vertices, we write $G - U$ for $G[V \setminus U]$. Let $F$ be a set of 2-element subsets of $V$. We write $G - F := (V, E \setminus F)$ and $G + F := (V, E \cup F)$.

A class of graphs that is closed under isomorphism is called a *graph property* and graphs in this class *have* this property. We call $G = (V, E)$ *edge-minimal* with a given graph property, if $G$ itself has this property, but for no $xy \in E$ the graph $G - \{xy\}$ does.

Graph $G = (V, E)$ is said to be *$H$-free* for a given graph $H$ if no induced subgraph of $G$ is isomorphic to $H$.

A *path* is a non-empty graph $P = (V, E)$ of the form

$$V = \{x_0, x_1, \ldots, x_k\} \qquad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$$

where all the $x_i$ are distinct. The vertices $x_0$ and $x_k$ are *linked* by $P$ and are called its *endpoints* and the path may be denoted by $x_0 \sim x_k$. The vertices $x_1, \ldots, x_{k-1}$ lie on the path $P$. The number of edges of a path is its *length*, and a path of length $k$ is denoted by $P_k$. Two paths are said to be *vertex disjoint*, if no vertex lies on both of them. Two paths are *edge disjoint* if they do not have a common edge.

The distance $d_G(x, y)$ in $G$ of two vertices $x, y$ is the length of a shortest path linking $x$ and $y$. The greatest distance between any two vertices in $G$ is the *diameter* of $G$, denoted by *$diam(G)$*.

A *cycle* is a non-empty graph $C = (V, E)$ of the form

$$V = \{x_0, x_1, \ldots, x_k\} \qquad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k, x_kx_1\} k \geqslant 2$$

where all the $x_i$ are distinct. The number of edges (which is equal to the number of vertices) of a cycle is its *length* and the cycle of length $k$ is denoted by $C_k$.

Given a graph $H$ we call the path $P$ an *$H$-path* if $P$ has length greater than 0 and meets $H$ exactly in its endpoints. In particular, the edge of any $H$-path of length 1 is never an edge of $H$. We call the cycle $C$ an *$H$-cycle* if $C$ has exactly one common vertex with $H$. Given two subsets of vertices $A, B$ we call $P$ an *$A$–$B$ path* if $P$ has one of its endpoints in $A$ and the second one in $B$, and none of the vertices from $A \cup B$ lie on $P$.

A non-empty graph $G = (V, E)$ is *connected* if any two vertices in $G$ are linked by a path in $G$. Graph is *disconnected* if it is not connected. $G$ is *$k$-connected* (for $k \in \mathbb{N}$) if $|G| > k$ and for every set $X \subseteq V$ with $|X| < k$ graph $G[V \setminus X]$ is connected. The greatest integer $k$ such that $G$ is $k$-connected is the *connectivity* $\kappa(G)$ of $G$. If $|G| > l$ and $G - F$ is connected for every set $F \subseteq E$ of fewer then $l$ edges, then $G$ is called *$l$-edge-connected.* The greatest integer $l$ such that $G$ is $l$-edge-connected is the *edge-connectivity* $\lambda(G)$ of $G$. The maximal $k$-connected ($l$-edge-connected) subgraphs of $G$ are called *$k$-connected components* (*$l$-edge-connected components*).

Given sets $A, B, C \subseteq V$ we call $C$ an *$(A, B)$-separator* if every $A$–$B$ path in $G$ contains a vertex from $C$. Any $F \subseteq E$ such that there is no $A$–$B$ path in $G - F$ is called an *$(A, B)$-edge-separator.*

The graph shown in Figure 1 is connected and 2-edge-connected but it is not 2-connected. It has two 2-connected components - $\{1, 2, 3\}$ and $\{2, 4, 5\}$.

A *coloring* of a graph $G = (V, E)$ is a map $c : V \to S$ such that $c(x) \neq c(y)$, whenever $xy \in E$. The elements of $S$ are *colors.* The smallest $k$ such that $G$

has coloring $c : V \to \{1, \ldots, k\}$ is the *chromatic number* of $G$ and is denoted by $\chi(G)$.

**Proposition 1** *If the graph $G$ is non-empty, then $\kappa(G) \leqslant \lambda(G) \leqslant \delta(G)$.*

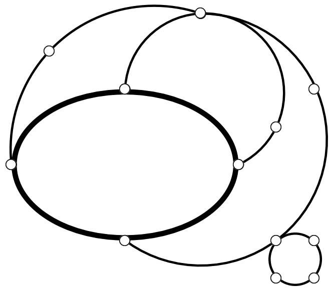
**Proposition 2** *A graph $G$ is connected if and only if its vertices may be enumerated as $v_1, v_2, \ldots, v_n$, so that $G[\{v_1, v_2, \ldots, v_i\}]$ is connected for every $i \leqslant n$.*

**Proposition 3** *A graph is 2-connected if and only if it can be constructed from a cycle by successively adding $H$-paths to the graphs $H$ which have already been constructed.*

Proofs of the above propositions can be found for example in [4].

**Proposition 4** *A graph is 2-edge-connected if and only if it can be constructed from a cycle by successively adding $H$-paths and $H$-cycles to the graphs $H = (V_H, E_H)$ which have already been constructed.*

Figure 2: Sample construction



**Proof:** Clearly, a cycle is 2-edge-connected, and when an $H$-path or $H$-cycle is added to a graph, it stays 2-edge-connected. Suppose that graph $G = (V, E)$ is 2-edge-connected. There are at least 3 vertices in $V$ and $G$ is connected, so $E$ is non-empty. Let $x, y$ be two vertices connected by an edge. Removing the edge $xy$ leaves the graph connected, so there exists a path joining vertices $x$ and $y$ in $G - \{xy\}$. This path and the edge $xy$ form a cycle $C$ in $G$. Let $H = (V_H, E_H)$ be a maximal subgraph of $G$ that can be constructed from $C$ by adding $H$-paths and $H$-cycles. Any edge $xy \in E$ that connects two vertices in $V_H$ is in $E_H$ – otherwise $xy$ would form an $H$-path, and the graph $H$ would not be maximal. So, $H$ is an induced subgraph of $G$. If $H \neq G$, then there exists a vertex $z \in G - H$. Graph $G$ is connected so there is a $z \sim x$ path in $G$. Let $vw$ be an edge on $z \sim x$ such that

$v \in G - H$ and $w \in H$. Graph $G - \{vw\}$ is connected, and there is a path $v \sim x$ in it. Let $u$ be the first vertex in $H$ on that path. If $u \neq w$, then $wv$ and $v \sim u$ form an $H$-path. Otherwise, $wv$ and $v \sim u$ form a cycle with exactly one vertex $u$ in $H$. In both cases, the construction can be extended, which contradicts the maximality of $H$. This proves that $G = H$. □

**Theorem 5** (Menger 1927) *Let $G = (V, E)$ be a graph and $A, B \subseteq V$ such that $A \cap B = \emptyset$. The following statements are true:*

1. *The minimum size of $(A, B)$-separator in $G$ is equal to the maximum number of vertex disjoint $A$–$B$ paths in $G$.*

2. *The minimum size of $(A, B)$-edge-separator in $G$ is equal to the maximum number of edge disjoint $A$–$B$ paths in $G$.*

3. *$G$ is $k$-connected if and only if it contains $k$ vertex disjoint paths between any two vertices.*

4. *$G$ is $l$-edge-connected if and only if it contains $l$ edge disjoint paths between any two vertices.*

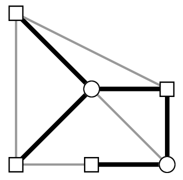
Several different proofs using various approaches can be found in [4].

## 2 Minimal connected spanning subgraphs and an off-line solution

We consider a problem of coloring a $k$-(edge)-connected spanning subgraph of a given graph $G = (V, E)$ with as few colors as possible. As the $k$-(edge)-connectivity is a *monotone* graph property (if $G$ has the property itself, then every supergraph on the same set of vertices does), we assume that $G$ is $k$-(edge)-connected. An algorithm solving the problem needs to construct a map $c : V \to S$ such that there exists a graph $H \subseteq G$ that is a spanning subgraph of $G$, it is $k$-(edge)-connected, and $c$ is a coloring of $H$. Graph $H$ does not have to be computed explicitly because having a map $c$ it is easy to compute an edge-maximal graph $H'$ for which $c$ is a vertex coloring. Edges of $H'$ are those edges of $G$ which join vertices with different colors in $c$. $H$ has to be a subgraph of $H'$ and from the monotonicity of $k$-(edge)-connectivity, $H'$ is $k$-(edge)-connected.

The first approach to solving the problem is to divide it into two subproblems: finding a $k$-(edge)-connected spanning subgraph $H$ and constructing a coloring of $H$. The $H$ we would seek for should minimize the value of $\chi(H)$. Obtaining $\chi(H)$ for a given graph $H$ is an $NP$-complete problem so we have to look for other criteria for finding a subgraph $H$. Clearly, the coloring of a supergraph

Figure 3: A graph with a coloring of a connected spanning subgraph



This graph has been colored using 2-colors (circles and squares). A connected spanning subgraph, for which the coloring is proper, exists. The edges of the subgraph are drawn with thicker lines.

induces the coloring of any of its subgraphs, and $H \subseteq H' \Rightarrow \chi(H) \leqslant \chi(H')$. This leads to a search for edge-minimal $k$-(edge)-connected spanning subgraphs of $G$.

Finding any edge-minimal $k$-(edge)-connected subgraph of a given graph $G$ is an easy task. Deciding whether a graph is $k$-edge-connected may be done in polynomial time for every $k$. This leads to a simple algorithm that removes edges from $G$ one by one if the resulting graph is still $k$-(edge)-connected. More subtle (and achieving better time complexities) algorithms can be found in [9] and [10].

The search for edge-minimal $k$-(edge)-connected subgraphs satisfying other desirable properties is usually $NP$-complete. For example, finding a $k$-(edge)-connected spanning subgraph minimizing the number of edges is $NP$-complete for $k \geqslant 2$ as described in [8]. The approximation algorithm for this problem is given in [3]. Finding a $k$-(edge)-connected spanning subgraph minimizing the maximal-degree is $NP$-complete [8]. For $k = 1$ the problem has an approximation to within one from optimal [7]. Approximation algorithms are also known for $k \geqslant 2$ [5].

If we were able to give a constant upper-bound on the chromatic number of the edge-minimal $k$-(edge)-connected subgraphs, we could use an algorithm which does not search for any special subgraphs and uses any of the edge-minimal $k$-(edge)-connected spanning subgraphs. Such an algorithm would achieve an approximation within constant to the optimal solution. If such an upper-bound does exist, it is at least $k+1$ (as $K_{k+1}$ is edge-minimal $k$-(edge)-connected and $\chi(K_{k+1}) = k+1$). We will now prove that such an upper-bound exists for $k \leqslant 2$ and equals $k+1$.

**Proposition 6** *Every edge-minimal connected graph is* 2*-colorable.*

**Proof:** Given a graph $G = (V, E)$ that is edge-minimal connected from Proposition 2 we obtain that $V = \{v_1, v_2, \ldots, v_n\}$ and $G[\{v_1, v_2, \ldots, v_i\}]$ is connected for every $i \leqslant n$. There is exactly one edge from $v_i$ to $\{v_1, v_2, \ldots, v_{i-1}\}$. If there were two such edges, one could be removed from $G$ and the resulting graph would also be connected, contradicting the edge-minimality of $G$. This leads to a simple 2-coloring algorithm for $G$. We set $c(v_1) := 1$ and give colors to other vertices in the increasing order. For each $2 \leqslant i \leqslant n$, the only neighbor of $v_i$ in $\{v_1, v_2, \ldots, v_{i-1}\}$ has already been given a color, and we can give the other one to $v_i$. □

**Proposition 7** *Every edge-minimal* 2*-(edge)-connected graph is* 3*-colorable.*

**Proof:** Propositions 3 and 4 give a inductive characterization of the 2-(edge)-connected graphs. In order to obtain 2-(edge)-connected, we can strengthen the requirements for $H$-paths added in these construction.

Graph $G$ is edge-minimal 2-(edge)-connected if it can be obtained by the construction as in Propositions 3, 4 in which $P_1$ was never used as an $H$-path. If $P_1$ was used in the construction, we could omit this step and finish the construction. The resulting graph would also be 2-(edge)-connected and it would be a spanning subgraph of $G$, contradicting the edge-minimality.

We can 3-color $G$ using the inductive characterization (without $H$-paths of length 1):

1. The starting cycle is 3-colorable.

2. When an $H$-path $P$ of length greater than 1 is added to graph $H$ then colors of its endpoints $x_1$ and $x_2$ are already set. Suppose that both of these colors are from $\{1, 2\}$. We can assign color 3 to any vertex $y$ inside $P$, and give alternating colors 1 and 2 on the paths $x_1 \sim y$ and $x_2 \sim y$ starting from $x_1$ and $x_2$ respectively.

3. When an $H$-cycle is added only one vertex is colored, and the rest of the cycle is a path which can be colored using the remaining two colors.

□

**Theorem 8** *A* $(k+1)$*-coloring of a* $k$*-(edge)-connected spanning subgraph can be efficiently computed for* $k \leqslant 2$*.*

**Proof:** Given a graph $G$, its edge-minimal $k$-(edge)-connected spanning subgraph can be computed as discussed earlier. Having this subgraph and following its construction (which can be easily obtained in quadratic time), the coloring can be established using the techniques from Proposition 6 and 7. □

## 3 On-line version of the problem

The on-line coloring of a graph can be viewed as a game between two players: Spoiler and Painter. The game is divided into rounds. In the first round Spoiler presents a graph and Painter gives a color to each vertex in this graph. In each following round Spoiler adds a new vertex $x$ to the graph already constructed and describes all edges joining $x$ and vertices from the previous rounds. Painter responds by giving $x$ a color. Once a Painter gives a color to $x$, it can never be changed. If an edge between $x$ and another vertex from the graph is not described by Spoiler, it can never appear in the future. Of course, Spoiler can connect $x$ with vertices added later on.

In the classical on-line graph coloring problem, the goal of the Painter is to have a correct coloring of the graph after every round, and to use as few colors as possible. Painter's strategy is usually measured by the *competitivity ratio* i.e. the number of colors he used divided by the chromatic number of the graph. In the general setting, there is no strategy for Painter that would guarantee a finite competitive ratio for all possible games. However, if some restrictions are imposed on the graphs that may be constructed by Spoiler, it often leads to games where Painter has an algorithm achieving a finite competitive ratio. Examples of such games can be found in [2] or [11].

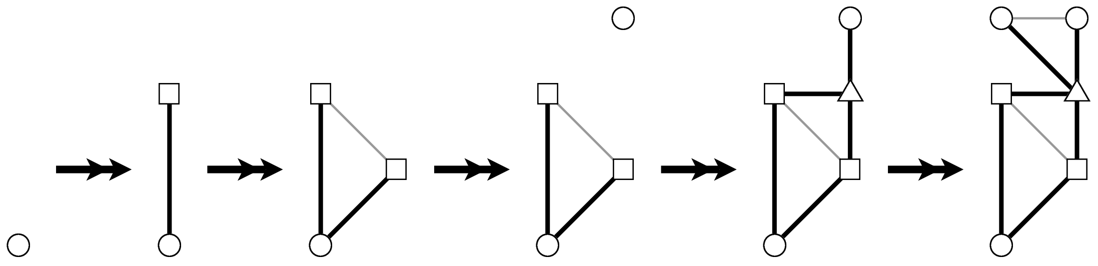
In our problem, Painter's task is to give colors to the vertices in such a manner, that whenever the resulting graph is $k$-(edge)-connected, there is a $k$-(edge)-connected spanning subgraph, which is correctly colored by Painter's coloring. The number $k$, and the kind of connectivity (vertex or edge) are parts of the problem specification. We will say that the task of the game is to maintain $k$-(edge)-connectivity, or simply call the game as *$k$-(edge)-connectivity game*. It is easy to check that playing for all $k$ simultaneously is equivalent to the classical problem of the on-line coloring.

The Spoiler-Painter game may be viewed as follows: Spoiler builds a graph as in the classical problem, but Painter constructs coloring of his own copy of the graph. Whenever Spoiler adds an edge to the graph, Painter decides whether to add it or not to his copy of the graph. Whenever Spoiler's graph is $k$-(edge)-connected, Painter's copy must have the same property. Even if Painter does not use this approach in his algorithm, we can compute the edge-maximal subgraph, for which his coloring is proper. We can treat this subgraph as a Painter's copy as described above.

The first trivial algorithm solving the on-line problem is the classical on-line coloring algorithm, which colors vertices in such a manner that all edges from Spoiler's graph are present in Painter's graph.

If Spoiler is able to play in such a way that his graph is $k$-(edge)-connected, and Painter's copy is not, then Painter is instantly defeated. We want to construct an algorithm for Painter that could not be defeated in this way. Such Painter's

Figure 4: Sample game



After 6 rounds Painter used 3 colors and maintained a connected spanning subgraph.

algorithm is *invincible*. We will give some rules which need to be respected by such an algorithm. Having these we will define an algorithm that is invincible, and is different from the trivial one (for which invincibility is easy to check).

**Lemma 9** *Assume that the task of the game is to maintain $k$-connectivity and Painter is invincible. If a set $K$ of $1 \leqslant i < k$ vertices is an $(A, B)$-separator in Painter's graph after Painter's move, then $K$ is an $(A, B)$-separator in Spoiler's graph as well.*

**Proof:** Suppose that at some point of the game, after Painter's move the Spoiler's graph is $G = (V, E)$, Painter's copy is $G' = (V, E') \subset G$, and $K \subseteq V$, $|K| = i$ is an $(A, B)$-separator in $G'$, but it is not in $G$. We will prove that Spoiler has a strategy to build a graph that will be $k$-connected, and Painter will not be able to keep his copy $k$-connected.

$K$ is an $(A, B)$-separator - this means that $G' - K$ is disconnected, and $A$ and $B$ are subsets of different components. Let $X, Y$ be two disjoint sets covering $V \setminus K$ such that $A \subseteq X$, $B \subseteq Y$, and there are no paths in $G'$ from $X$ to $Y$. Spoiler's strategy is as follows and is based on the fact that $G - K$ is connected:

1. Add $k$ new vertices $X'$, which are adjacent to all vertices in $X \cup X' \cup K$.

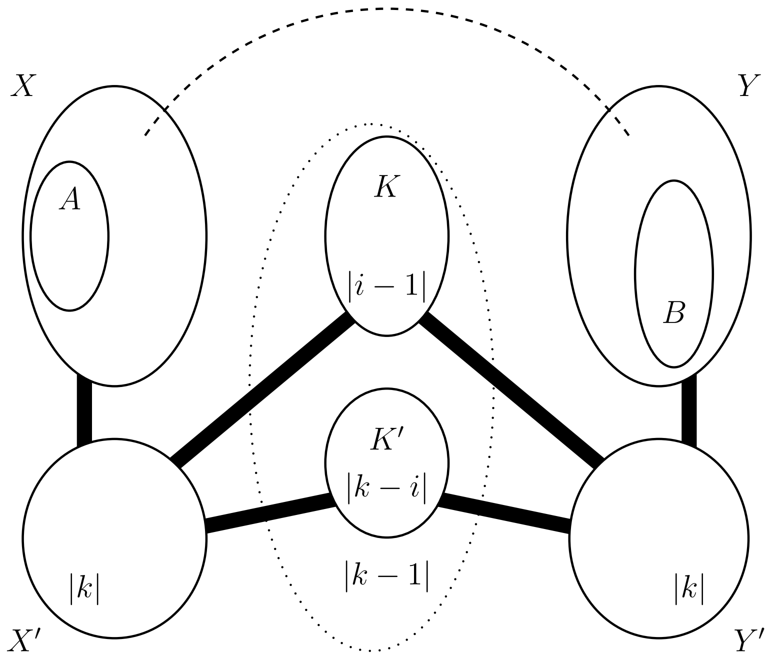
2. Add $k$ new vertices $Y'$, which are adjacent to all vertices in $Y \cup Y' \cup K$.

3. Add $k - i$ new vertices $K'$, which are adjacent to all vertices in $X' \cup Y'$.

Clearly Painter's copy cannot be $k$-connected after this construction. Removal of $K \cup K'$, which is of size $k - 1$, leaves his graph disconnected.

Spoiler's graph, on the contrary, is $k$-connected. We will prove this by using Menger's Theorem 5. We need to construct $k$ vertex-disjoint paths between any two vertices.

Figure 5: Spoiler's strategy



- For any two vertices in $X \cup X' \cup K \cup K'$ the paths are easily constructed by using $k$ vertices from $X'$ which are connected to all vertices in the set.
- The same holds for $Y \cup Y' \cup K \cup K'$.
- The only case left for discussion is one vertex in $X \cup X'$ and the other in $Y \cup Y'$. In this case, one path is obtained from the fact that $G - K$ is connected, and remaining $k-1$ can be constructed using vertices from $K \cup K'$.

After applying this strategy, Painter is defeated. This contradicts the invincibility of Painter's algorithm. □

**Lemma 10** *Assume that the task of the game is to maintain $l$-edge-connectivity and Painter is invincible. If a set $L$ of $1 \leqslant i < l$ edges is an $(A,B)$-separator in Painter's graph after Painter's move, then $L$ is an $(A,B)$-separator in Spoiler's graph as well.*

**Proof:** This proof is very similar to the previous one. Let $L$ be an $i$-element $(A,B)$-separator in $G'$. $G' - L$ is disconnected and $A$ and $B$ are subsets of different components. Let $X, Y$ be two disjoint sets covering $V$ such that $A \subseteq X$,

$B \subseteq Y$ and there are no paths in $G'$ from $X$ to $Y$. Spoiler's strategy is now as follow:

1. Add $l$ new vertices $X'$, which are adjacent to all vertices in $X \cup X'$.
2. Add $l$ new vertices $Y'$, which are adjacent to all vertices in $Y \cup Y'$.
3. Add a vertex $z$, which is adjacent to all vertices in $X \cup X'$ and $l-i$ vertices in $Y'$.

Again, Painter is unable to obtain $l$-edge-connectivity. The set of edges between $z$ and $Y'$, together with edges from $L$, form an $(l-1)$-element $(A, B)$-separator. The $l$-edge-connectivity of Spoiler's graph is easy to check. □

**Corollary 11** *If the task of the game is to maintain $k$-connectivity (or $l$-edge-connectivity) and Painter is invincible, then for each $1 \leqslant i < k$ $(1 \leqslant i < l)$ and $i$ vertex disjoint (edge disjoint) $A$–$B$ paths in Spoiler's graph, there are $i$ such $A$–$B$ paths in Painter's graph.*

**Proof:** The proof using Menger's Theorem 5 and Lemmas 9 and 10 is straightforward. □

This corollary gives us strong bounds on how Painter may play, and allows us to analyze his possible strategies. We are also able to introduce the first on-line algorithm trying to solve the problem. The algorithm colors vertices with natural numbers and colors the newly added vertex with the smallest number such that Corollary 11 is satisfied. We will call this algorithm *First-Fit Algorithm* or $FFA$ for short. We will analyze $FFA$ both in general setting and when various restrictions are imposed.

**Theorem 12** *$FFA$ is invincible.*

**Proof:** What we need to prove is that at each step, when Spoiler adds a new vertex $x$, the color given to $x$ by $FFA$ preserves the property imposed on $G'$ by Corollary 11.

Suppose that the task of the game is to maintain $k$-connectivity. Let $P$ be a set of $1 \leqslant i < k$ vertex disjoint paths in $G + x$ having one of the endpoints in $A$ and the second one in $B$. In the proof $A$ and $B$ should be treated as multisets of endpoints of the paths. If $x$ does not lie on any path from $P$, then the same set $P$ is present in $G$ and the thesis holds by induction.

If $x$ is an endpoint of one of the paths $P_x \in P$, then without loss of generality, we can suppose that $x \in A$ and $x_B$ is the neighbor of $x$ in $P_x$. If $x_B \notin B$, we can substitute $P_x$ in $P$ by $P_x - x$. We obtain an $i$ element set of paths between

$A \setminus \{x\} \cup \{x_B\}$ and $B$ in the graph $G$. Painter using $FFA$ has maintained a corresponding set of $i$ paths in his graph and he can extend the path ending in $x_B$ with the edge $x_B x$. If $x_B \in B$ the situation is even simpler – adding the edge $x_B x$ to the Painter's graph solves both cases. This requires giving $x$ a color different from the color of $x_B$, but enables Painter to continue with $FFA$ algorithm.

If $x$ is not an endpoint of any of the paths in $P$, then it lies on $P_x \in P$, $a \in A$ and $b \in B$ are the endpoints of $P_x = a \sim x \sim b$. As proved earlier $FFA$ can maintain $i$ paths between $A \setminus \{a\} \cup \{x\}$ and $B$ (as well as between $B \setminus \{b\} \cup \{x\}$ and $A$). Using Menger's Theorem 5 we obtain that there is no set with less then $i$ elements separating:

1. $A$ from $B$ in $G + x$.
2. $A \setminus \{a\} \cup \{x\}$ from $B$ in $G' + x$.
3. $B \setminus \{b\} \cup \{x\}$ from $A$ in $G' + x$.

We need to prove that there is no such a set separating $A$ from $B$ in $G' + x$. Suppose that there is such a set $C$. If $x \notin C$, then $C$ separates 2 or 3, so $x \in C$. This means that $C \setminus \{x\}$, having $i - 2$ elements, was separating $A \setminus \{a\}$ from $B \setminus \{b\}$ in $G'$, which is impossible because Corollary 11 was satisfied for $G$ and $G'$ and $P \setminus \{P_x\}$ was a set of $i - 1$ paths between $A \setminus \{a\}$ and $B \setminus \{b\}$ in $G$.

The same proof with minor changes works for $l$-edge-connectivity game. $\square$

## 3.1 Graph is $k$-(edge)-connected during the whole game

If no restrictions are applied to the game, Spoiler has a strategy to force Painter to use any number of colors to maintain connectivity of a 2-colorable graph. This strategy will be discussed later on. This is why we introduce limitations on how Spoiler is allowed to play. The first, natural restriction is to force Spoiler to start from $K_{k+1}$ and keep the graph $k$-(edge)-connected all the time.

We prove a simple fact which will allow us to examine the behavior of $FFA$ in this setting.

**Lemma 13** *Given $k$-connected graph $G$ and a new vertex $x$ with at least $k$ neighbors in $G$, graph $G + x$ is $k$-connected.*

**Proof:** Let $X$ be a set of $k$ different neighbors of $x$ in $G$. To prove $k$-connectivity of $G + x$, we will show that removing any $k - 1$ vertices $K$ leaves the graph connected. If $x$ was one of the removed vertices, then the resulting graph is $G$ with $k - 2$ vertices removed, and it is connected because $G$ was $k$-connected. Otherwise the resulting graph is $(G - K) + x$. $G$ was $k$-connected so $G - K$ is connected and because $|K| < |X|$ there is at least one vertex $y$ both in $X$ and outside of $K$. The graph $(G-K)+x$ is connected because $x$ is linked with $y$. $\Box$

**Corollary 14** *$FFA$ uses exactly $k+1$ colors, when Spoiler starts the game with $K_{k+1}$ and keeps the graph $k$-(edge)-connected during the whole game.*

**Proof:** In this setting there is no difference between the connectivity game and the edge-connectivity game. Even if the task of the game is to maintain $k$-edge-connectivity, the graph stays $k$-connected all the time. This can be proved using induction on the number of rounds in the game. Clearly, $K_{k+1}$ is $k$-connected, and when Spoiler adds a vertex to the graph, he is obliged to keep the graph $k$-edge-connected. This means that the new vertex needs to have the degree at least $k$, and using Lemma 13 we obtain $k$-connectivity of the resulting graph.

$FFA$ colors the starting graph $K_{k+1}$ using $k+1$ colors and when a new vertex is added, then the algorithm uses the smallest number, such that the new vertex has $k$ neighbors with color different from the chosen one. Clearly, $FFA$ will never use color $k+2$, because $k$ neighbors can have only $k$ different colors. $\Box$

This result shows that the restriction chosen significantly limits Spoiler. In fact, he is not able to build all possible $k$-edge-connected graphs. Even a $C_4$ cannot be achieved in a 2-connectivity game.

## 3.2 On-line coloring of a connected subgraph

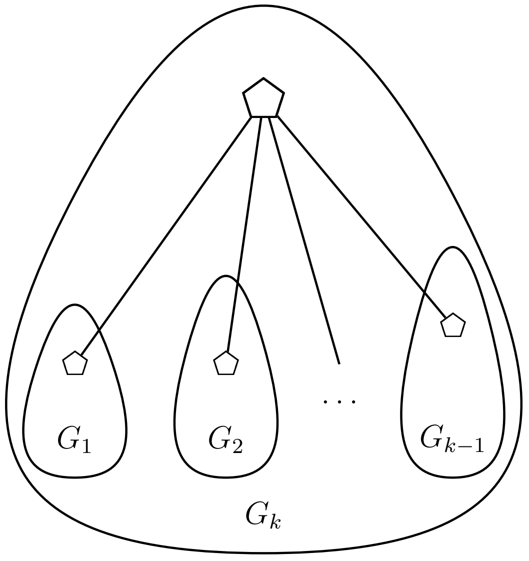
We will now discuss the connectivity game without any restrictions on how Spoiler is supposed to play. Amazingly when analyzing this setting we will also find characterization of some interesting restrictions of the game.

Firstly, we show a strategy for Spoiler that forces $FFA$ algorithm to use any number of colors when Spoiler's graph is 2-colorable. This strategy is constructed recursively - $G_1$ is a graph with one vertex, and in order to construct a graph $G_k$ we construct vertex disjoint graphs $G_1, G_2, \dots G_{k-1}$ without any edges between them, and then add a new vertex adjacent to vertex colored to $i$ in each $G_i$. This construction is shown in Figure 6.

Clearly, $FFA$ gives color $k$ to the last vertex in $G_k$. No lower color $i$ can be used because connectivity between the new vertex and $G_i$ would be lost.

The same recursive strategy may be used against any other Painter's algorithm. To force Painter to use $k$ different colors, Spoiler constructs $k-1$ graphs with no edges between them. The construction of the $i$-th graph is finished when Painter introduces the $i$-th color. To finish the construction, Spoiler adds a new vertex adjacent to $k-1$ vertices with different colors placed in different components. Painter has to introduce new color for this vertex. Any graph obtained by this construction will be denoted by $G^*_k$.

Figure 6: Strategy $G_k$



Both $G_k$ and any possible $G_k^*$ are edge-minimal connected. Removing any of the edges at any step of the construction would result in a disconnected graph. Using Proposition 6, we obtain the 2-colorability of $G_k$ and $G_k^*$.

We have constructed a strategy for Spoiler to play against any Painter's algorithm and to force him to use $k$ colors. Next lemma shows that there is no other strategy that forces $FFA$ to use $k$ colors.

**Lemma 15** *In a connectivity game, for each $k \geqslant 2$, if Spoiler's graph is $G_k$-free and Painter applies $FFA$, then Painter uses less than $k$ colors.*

**Proof:** We prove this by induction on $k$. For $k = 2$ the thesis is trivial. Suppose that $FFA$ has used color $k$ for a vertex $x_k$. This means that in Painter's copy of the graph, for each $i = 1, 2, \dots, k-1$ there is a vertex $x_i$ colored $i$ that is adjacent to $x_k$. Otherwise $x_k$ would be given the missing color. Moreover, we can select $x_i$'s so that they all lie in different components. If one of the colors was not represented in this way, $FFA$ would use this color for $x_k$ and all connectivities would be satisfied. By induction, there is an induced subgraph $G_i$ in the component of $x_i$, so these $G_i$ together with $x_k$ form $G_k$. $\square$

An interesting class of possible restrictions of the game is obtained by forbidding some structures to appear in the Spoiler's graph. When the graph $H$ is forbidden, then we will call the game to be *$H$-free*. For example, we may analyze a game in which Spoiler's graphs have to be $K_3$-free all the time. This is $K_3$-free connectivity game.

The construction of $G_k$ and Lemma 15 give us a simple method of checking how well $FFA$ is doing in an $H$-free game. The only thing that needs to be checked is for which $k$ graphs $G_k$ are $H$-free. When $H = K_3$ is discussed, it is easy to check that for all $k \in \mathbb{N}$ the graph $G_k$ is $K_3$-free. This leads to a corollary that Spoiler can force $FFA$ to use any number of colors in a $K_3$-free connectivity game. Moreover, any possible $G_k^*$ is $K_3$-free, so any Painter's algorithm can be forced to use any number of colors as well.

$P_m$-free games are especially interesting for us. For example, graphs with diameter limited up to $m-1$ belong to the class of $P_m$-free graphs.

**Lemma 16** *$FFA$ uses at most $\lfloor \frac{m+1}{2} \rfloor$ colors in a $P_m$-free connectivity game.*

**Proof:** We will prove that $G_k$ has an induced subgraph isomorphic to $P_{2k-3}$. Let us introduce two sequences: $p_i$ is the length of the longest induced path in $G_i$; $h_i$ is the length of the longest induced path in $G_i$ having one of its endpoints at the vertex colored $i$ by $FFA$. The following recursive formulas emerge almost automatically from the construction of $G_k$.

$$\begin{aligned}
h_1 &= 0 \\
p_1 &= 0 \\
p_2 &= 1 \\
h_{i+1} &= max(h_1, h_2, \ldots, h_i) + 1 \\
p_{i+2} &= max(p_1, p_2, \ldots, p_{i+1}, h_1 + h_2 + 2, h_1 + h_3 + 2, \ldots, h_i + h_{i+1} + 2)
\end{aligned}$$

which are easy to solve:

$$\begin{aligned}
h_{i+1} &= h_i + 1 = i \\
p_{i+2} &= h_i + h_{i+1} + 2 = 2 \cdot i + 1
\end{aligned}$$

So $p_i = 2 \cdot i - 3$ for $i \geqslant 2$. This proves that $P_m$-free graphs are $G_{\lfloor \frac{m+3}{2} \rfloor}$-free and by Lemma 15 $FFA$ uses at most $\lfloor \frac{m+1}{2} \rfloor$ colors to color any of them. $\square$

**Lemma 17** *Spoiler can force every Painter's algorithm to use $\lfloor \frac{m+1}{2} \rfloor$ colors in a $P_m$-free connectivity game.*

**Proof:** Using the same approach as in the previous proof, we can inductively show that for $i \geqslant 2$,

$$\begin{aligned} h_i(G_i^*) &\leqslant i-1 \\ p(G_i^*) &\leqslant 2 \cdot i - 3; i \geqslant 2 \end{aligned}$$

where $h_i(G_i^*)$ is the length of the longest induced path in $G_i^*$ having one of its endpoints in the vertex colored $i$, and $p(G)$ is the length of the longest path in $G$.

Having these inequalities, we can state that the construction of any $G^*_{\lfloor \frac{m+1}{2} \rfloor}$ is possible in the class of $P_m$-free graphs. This proves that Spoiler can force Painter who uses any valid algorithm to use at least $\lfloor \frac{m+1}{2} \rfloor$ colors in a $P_m$-free connectivity game. □

Lemmas 16 and 17 show that $FFA$ is an optimal algorithm in $P_m$-free connectivity games.

## Further Research

It seems, that a result similar to Theorem 8 may be true for every $k$. For $k = 3$ there is an inductive characterization of 3-connected graphs by Tutte [4], however it is not that easy to use. There is no characterization known for $k > 3$. Thus, proving this conjecture would require techniques different from the one used in the proof of Propositions 6 and 7.

$FFA$'s behavior has been thoroughly examined only for the connectivity game. In a 2-edge-connectivity game it is still quite easy to follow this algorithm, but for higher $k$ it is troublesome.

## Thanks

I would like to thank my friends and colleagues from the „Underground Seminar" for inspiring my research and for introducing me to the techniques which occurred to be crucial in my work.

## References

[1] M. Cai, *The number of vertices of degree k in a minimally k-edge-connected graph*, J. Combin. Theory Ser. B, 58 (1993), pp. 225–239.

[2] I. Cieślik, *On-line Graph Coloring*, Ph.D. thesis, Jagiellonian University (2006).

[3] J. Cheriyan and R. Thurimella, *Approximating minimum-size k-connected spanning subgraphs via matching*, SIAM J. Comput., 30 (2000), pp. 528–560.

[4] R. Diestel, *Graph Theory*, Springer-Verlag, New York, 2005.

[5] T. Feder, R. Motwani and A. Zhu, *k-connected spanning subgraphs of low degree*, Electronic Colloquium on Computational Complexity, Report 41 (2006).

[6] A. Frank, T. Ibaraki, and H. Nagamochi, *On sparse subgraphs preserving connectivity properties*, J. Graph Theory, 17 (1993), pp. 275–281.

[7] M. Furer and B. Raghavachari, *Approximating the Minimum Degree Spanning Tree to within One from the Optimal Degree*, Proc. of the Third Annual ACM–SIAM Symposium on Discrete Algorithms, Orlando, FL (1992), pp. 317–324.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[9] X. Han, P. Kelsen, V. Ramachandran, and R. Tarjan, *Computing minimal spanning subgraphs in linear time*, SIAM J. Comput., 24 (1995), pp. 1332–1358.

[10] H. Nagamochi and T. Ibaraki, *A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph*, Algorithmica, 7 (1992), pp. 583–596.

[11] M. Ślusarek, *Optimal on-line coloring of circular arc graphs*, RAIRO Theoretical Informatics and Applications, 29 (1995), pp. 423–429.