Jagiellonian University

Mathematics and Computer Science Faculty

Computer Science Institute

Dimension online of upgrowing orders

Piotr Micek

Master Thesis
Advisor: Professor Paweł Idziak, Ph. D.

Kraków, June 2004

Abstract

We analyze the adaptive coloring online problem for the upgrowing orders. We consider this problem as a two-person game. One person builds an order one point at a time. The other person responds by coloring the new point according to strictly determined rules. This problem is equivalent to certain variant of the dimension online problem. The adaptive coloring problem can be also applied to the tasks scheduling online problem. Until now no non-trivial lower bound for these problems has been known. This paper contains a proof that for orders of width at most d any algorithm can be forced to use almost $2 \cdot d$ colors - in other words the lower bound for adaptive coloring online problem of upgrowing orders of width at most d is almost $2 \cdot d$ colors.

Our considerations are motivated by the scheduling problem in a certain parallel environment. Tasks to be done are coming online but it is unknown how long their execution will take and when the next task appears. Tasks may be self-dependent. Hence, with the new task x comes a (possibly empty) list of tasks, which have to be executed before the execution of the new task x. At the moment of appearance of the new task we must immediately schedule a processor or group of processors which will execute this task. In such environment we consider the optimal scheduling problem. It means that tasks have to be executed as fast as it is possible but we also want to use the least number of processors. The number of used processors will be compared with the maximal number of independent tasks.

All online problems considered in this paper can be viewed as two-person game. We call the players the Algorithm and the Spoiler. The Algorithm represents an online algorithm and the Spoiler represents an adaptive adversary. The game is played in rounds. During each round the Spoiler introduces a new point x to the order and describes comparabilities between x and points from previous rounds. The Algorithm responds by assigning x to a chain. The most important thing in online problems is that the previous Algorithm's moves (decisions) restrict his actual possibilities.

Kierstead, McNulty and Trotter [2] have investigated *dimension-online*(d) problem. In the corresponding game the Spoiler builds an online order of given width d while the Algorithm maintains its online realizer i.e., the set of linear extensions that intersect to binary relation of partial ordering. During each round the Spoiler adds point x to $\mathcal{P}$ (the poset of the game). Now the Algorithm has to fix point x in all extensions already created. He may of course create new extensions but he may not change relations between points from previous rounds in extensions already created. *Dimension-online*(d) is

the largest integer so that the Spoiler has strategy generating poset of width at most d which forces the Algorithm to use *dimension-online*(d) linear extensions. Note that by game theoretic duality we may as well define *dimension-online*(d) as the least integer so that there is an algorithm that never uses more extensions for posets of width at most d. Since the dimension of an order never exceeds its width it is natural to compare the dimension-online of considered posets to their width. The main negative result of [2] is:

Theorem 1 (Kierstead,McNulty,Trotter [2]) *For each* $d \geq 3$

$$\textit{dim-online}(d) = \infty.$$

This result closes considerations under *dimension-online*(d). Felsner[1] introduced the *dimension-online-upgrowing*(d) problem. In comparison with the *dimension-online*(d) problem the moves (possibilities) of the Spoiler are restricted to points which are maximal at the moment they are added to the poset. We can say that new points are dropped from the top. In [1] Felsner proved also an upper bound for this problem:

Theorem 2 (Felsner [1])

$$\textit{dim-online-upgrowing}(d) \leq \binom{d+1}{2}$$

Further in paper [1] an *adaptive-coloring-online-upgrowing*(d) problem (shortly - *adaptive*(d)) is defined. Let's describe the game corresponding to this problem. The Spoiler builds poset online upgrowing. During each round the Spoiler adds to the poset point x which is maximal in the current poset of the game. The Algorithm colors the new point x with some, non-empty set of colors. During the coloring of the new point he may remove (but never add) colors from other points. Following conditions must be respected:

- each point is colored by a non-empty set of colors
- during each round, for each color c the set of points colored by c form a chain

The value *adaptive*(d) is defined as the largest integer so that the Spoiler has a strategy forcing the Algorithm to use *adaptive*(d) colors. Equivalently it is the least integer so that there is an algorithm that never uses more colors. The paper [1] contains a proof of Theorem 3 (corrected by Kloch[3]).

Theorem 3 (Felsner [1],Kloch [3])

$$\textit{adaptive}(d) = \textit{dimension-online-upgrowing}(d)$$

Consider the scheduling problem described at the beginning of this paper. The value *scheduling-online*(d) is defined as the minimal number of processors necessary to execute in optimal time any group of tasks arriving online, where d is the maximal number of tasks independent in this group.

Scheduling-online(d) and *adaptive*(d) turn out to be equivalent. Tasks are treated as the points of a poset presented online in an up-growing way. Dependencies between the tasks are the edges of the poset. Processors are viewed as colors. For each processor tasks which were executed or scheduled to it must form a chain. Scheduling task y to processor p which has been executing task x independent with y we would not be sure whether processor p would be free at the time task y could be executed; so there would be some delay and this is not allowed.

Theorem 4

$$\textit{scheduling-online}(d) = \textit{adaptive}(d) = \textit{dim-online-upgrowing}(d)$$

Hence, all these three problems are equivalent. In the rest of this paper we deal with the *adaptive*(d) problem. As it was already mentioned the following upper bound was proved in [1] because the algorithm for all online posets can be as well applied to this particular upgrowing situation.

$$d \leq \textit{adaptive}(d) \leq \binom{d+1}{2} \tag{1}$$

The lower bound is trivial. It suffices for Bob to put antichain of d points. Until now no better result has been known. First we will give few definitions.

Definition 5 *By a poset online we mean a sequence (P_n) of posets in which P_{n+1} of P_n and a new point. This sequence starts with an empty poset.*

Definition 6 *By a poset upgrowing we mean poset online in which new points must be maximal in the poset at the moment they are added.*

Definition 7 *A multicoloring of the poset is a coloring of each point of the poset by a non-empty set of colors. For each color set of points containing that color has to be a chain.*

Definition 8 *By an adaptive coloring online of poset upgrowing we mean sequence of multicolorings of consecutively received posets of the poset online upgrowing. Consecutive multicoloring must be integrated with previous ones. This means that for each point p and for each multicoloring M_j in which p occurs $M_{j+1}(p) \subseteq M_j(p)$.*

Definition 9 *By an attack on point x we mean putting a new single point only above x.*

In our considerations we will need the following easy combinatorial Lemma.

Lemma 10 *Suppose s balls are distributed into b boxes, i.e., $s = s_1 + \ldots + s_b$. If $s_1 \leq s_2 \leq \ldots \leq s_b$ then for any $m \leq b$ we have*

$$s_1 + \ldots + s_m \leq \frac{m \cdot s}{b}.$$

Now we are ready to state the main theorem of this paper.

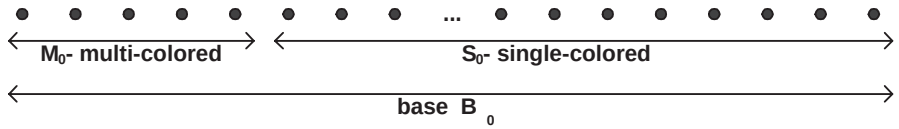
Theorem 11 *There is no online algorithm for adaptive coloring of upgrowing posets with competitive ratio strictly smaller than 2, i.e. for $c < 2$ there is no algorithm that uses only $c \cdot d$ colors, where d is the width of the poset.*

This theorem easily follows from the following Lemma.

Lemma 12 *For each $\varepsilon > 0$ and almost all $d \in \mathbb{N}$ the Spoiler has a strategy that builds a poset P of width d, and eventually forces all possible algorithms to use at least $(2 - \varepsilon)d$ colors on P.*

Proof. Fix $0 < \varepsilon < 2$ and suppose $d \geq 256/\varepsilon^6$. Let $c = \lfloor (2 - \varepsilon)d \rfloor$. Now our Lemma will be shown by describing a strategy for the Spoiler to force all possible algorithms to use at least $c + 1$ colors. In the rest of our proof we denote by $c(x)$ the set of colors assigned (by the Algorithm) to the point x of the poset $\mathcal{P}$ and $c(X) = \bigcup_{x \in X} c(x)$ whenever $X \subseteq \mathcal{P}$.

The Spoiler will act in rounds, consecutively building levels L_i of the poset by adding new points to L_{i+1} that cover some of those from L_i. The set of the points from L_i that are to be covered by the Spoiler is called the base and denoted by B_i. Put L_0 to be an antichain consisting of d elements and $B_0 := L_0$.



Suppose L_i and B_i are already constructed by the Spoiler and let $b_i := |B_i|$. The Algorithm responded by coloring the points of L_i but the rest of the Spoiler strategy depends only on how points of B_i are colored. In particular B_i decomposes into a sum $B_i = M_i \cup S_i$. $M_i = \{x \in B_i : |c(x)| > 1\}$ is the set of multicolored points of B_i and S_i is the set of points in B_i that were colored by the Algorithm with a single color. Let $m_i := |M_i|$ and $s_i := |S_i|$.

During the entire game the Algorithm has only c colors to use. Note that on the level L_0 the Algorithm is not restricted in using colors, in particular, it can use up to $c_0 := c$ colors to color the set B_0. However moving up to the next level some of colors are going to be blocked by the Spoiler, so that they cannot be used by the Algorithm any more. In fact we are interested in the strictly decreasing sequence of numbers defined by

$$\begin{array}{rcl} c_0 & = & c \\ c_{i+1} & = & c_i - 2\lfloor\sqrt{b_i}\rfloor + \frac{2b_i}{2b_i - c_i} \end{array} \tag{2}$$

We will show that:

$$\begin{array}{c} c_i \text{ bounds from above the number of colors the Algorithm} \\ \text{can use to color points from } B_i \text{ and their succesors} \end{array} \tag{3}$$

Obviously, we can show (3) only after presenting the construction used by the Spoiler. However to present this construction we will need that

$$c_i \leq 2b_i - 2\sqrt{b_i} \tag{4}$$

Not going into details how the sets B_i's are obtained we note now that their sizes are given by

$$\begin{array}{rcl} b_0 & = & d \\ b_{i+1} & = & b_i - \lfloor\sqrt{b_i}\rfloor \end{array} \tag{5}$$

Now the fact that the inequality (4) is kept while going to the next level follows immediately by comparing the leftmost and the rightmost value in the following claim.

Claim For each $i \geq 0$ we have

$$2b_i - c_i \geq \frac{\varepsilon}{2} b_i + 2\sqrt{b_0} \geq 2\sqrt{b_i}$$

Proof. To see the second inequality note that $b_{i+1} = b_i - \lfloor\sqrt{b_i}\rfloor \leq b_i$ immediately gives $b_i \leq b_0$. To see the first inequality we induct on i. For $i = 0$ we recall out starting assumption $c_0 \leq (2 - \varepsilon)b_0$ to get

$$2b_0 - c_0 \geq 2b_0 - (2 - \varepsilon)b_0 = \varepsilon b_0 \tag{6}$$

Our initial choice of $b_0 = d$ tells us that $b_0 \geq \frac{256}{\varepsilon^6}$. Consequently for $\varepsilon \leq 2$ we have

$$\varepsilon b_0 \geq \frac{\varepsilon}{2} b_0 + 2\sqrt{b_0},$$

which together with (6) gives

$$2b_0 - c_0 \geq \frac{\varepsilon}{2} b_0 + 2\sqrt{b_0},$$

as required.

For the induction step we first observe that for $b_0 \geq \frac{256}{\varepsilon^6}$ the polynomial $\frac{\varepsilon^2}{8}x^2 - 2x + \frac{\varepsilon}{2}\sqrt{b_0}$ takes nonnegative values. Consequently for $x = \sqrt{b_i}$ we get

$$\frac{2b_i}{\frac{\varepsilon}{2}b_i + 2\sqrt{b_0}} \leq \frac{\varepsilon}{4}\sqrt{b_i}. \tag{7}$$

Since $x \leq 2\lfloor x \rfloor$ for $x \geq 1$ we have $\frac{\varepsilon}{4}\sqrt{b_i} \leq \frac{\varepsilon}{2}\lfloor\sqrt{b_i}\rfloor$ and we can rewrite (7) to

$$\frac{2b_i}{\frac{\varepsilon}{2}b_i + 2\sqrt{b_0}} \leq \frac{\varepsilon}{2}\lfloor\sqrt{b_i}\rfloor. \tag{8}$$

Now we can bound $2b_{i+1} - c_{i+1}$ from below as follows:

$$\begin{aligned}
2b_{i+1} - c_{i+1} &= 2(b_i - \lfloor\sqrt{b_i}\rfloor) - (c_i - 2\sqrt{b_i} + \frac{2b_i}{2b_i - c_i}) \\
&= (2b_i - c_i) - \frac{2b_i}{2b_i - c_i} \\
&\geq \frac{\varepsilon}{2}b_i + 2\sqrt{b_0} - \frac{2b_i}{\frac{\varepsilon}{2}b_i + 2\sqrt{b_0}} \\
&\geq \frac{\varepsilon}{2}b_i + 2\sqrt{b_0} - \frac{\varepsilon}{2}\lfloor\sqrt{b_i}\rfloor \\
&= \frac{\varepsilon}{2}b_{i+1} + 2\sqrt{b_0},
\end{aligned}$$

where the last inequality in the above display is obtained from (8). □

Note that since each point in B_i gets at least one color we get the followings upper bound on the size of M_i:

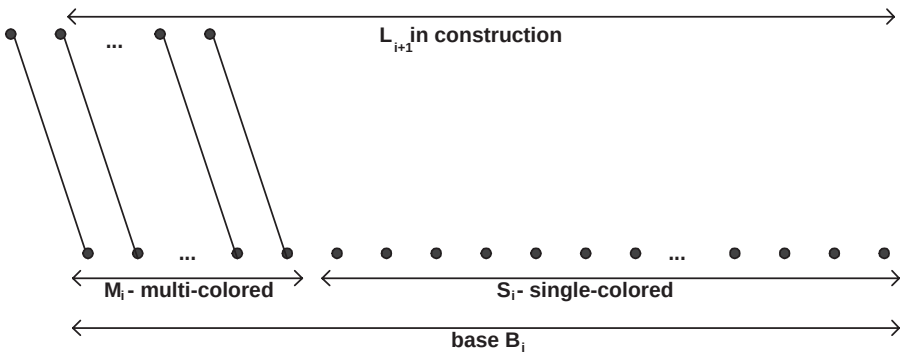
$$m_i \leq c_i - b_i. \tag{9}$$

Knowing that $m_i + s_i = b_i$ we immediately get the lower bound on the size of S_i:

$$s_i \geq 2b_i - c_i. \tag{10}$$

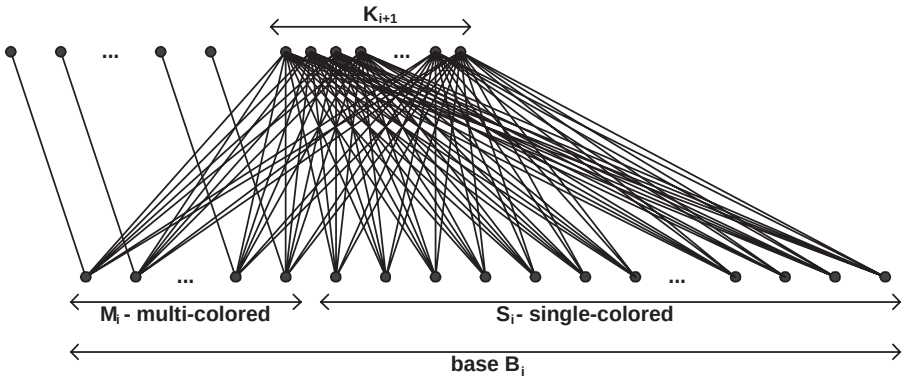
Again suppose L_i and B_i are already constructed and colored. The next level L_{i+1} is to consist of points that cover some of the ones from B_i.

- First each point x from M_i is attacked, i.e., it gets a succesor that covers only x. Till now L_{i+1} has exactly m_i points.



- Now the Spoiler puts into L_{i+1} an antichain K_{i+1} consisting of $\lfloor\sqrt{b_i}\rfloor$ points each of which lies over all of the points from B_i. Now L_{i+1} has $m_i + \lfloor\sqrt{b_i}\rfloor$ points, which by (9) and (4) is bounded by

$$m_i + \left\lfloor \sqrt{b_i} \right\rfloor \leq c_i - b_i + \sqrt{b_i} \leq 2b_i - 2\sqrt{b_i} - b_i + \sqrt{b_i} \leq b_i. \qquad (11)$$



Spoiler's strategy will eventually build L_{i+1} with exactly b_i points. This is to be accomplished by adding exactly $b_i - (m_i + \lfloor\sqrt{b_i}\rfloor)$ new points, but the way the Spoiler adds them depends on how points in K_{i+1} are colored.

- After the Algorithm colors all points from K_{i+1} the Spoiler lists them so that

$$\begin{gathered}\text{for } x, y \in K_{i+1} \text{ the point } x \text{ preceeds } y \text{ in this list} \\ \text{if } |C(x) \cap C(S_i)| \leq |C(y) \cap C(S_i)|\end{gathered} \qquad (12)$$

For $x \in K_{i+1}$, put

$$U(x) = \{s \in S_i : c(s) \subseteq c(x)\}, \tag{13}$$

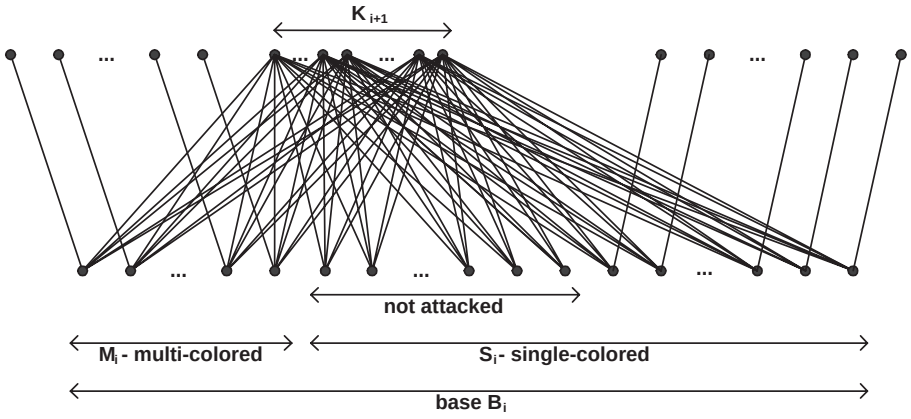
i.e., $U(x)$ consists of those points b from the base that are single-colored and the color of b is used by the Algorithm to color x. Note that, even if $c(x)$ will be changed in the future the set $U(x)$ will remain the same.

- Next, the Spoiler creates $A_{i+1} = \emptyset$ and than he follows the order of the list defined by (12) and adds new points into L_{i+1} up to the moment when the size L_{i+1} riches b_i. This means that once $b_i - (m_i + \lfloor\sqrt{b_i}\rfloor)$ are added to L_{i+1} by this subroutine, the next round is started.

  However, while $|L_{i+1}| < b_i$ the Spoiler consider consecutive point $x \in K_{i+1}$, adds it to A_{i+1} and adds to L_{i+1} a new point that covers only b (an attack on b) for each $b \in U(x)$. Remember that since the guard

$$|L_{i+1}| \leq b_i \tag{14}$$

  is active, it may happen that for some $x \in K_{i+1}$ only a part of points from $U(x)$ are covered this way. Note also that, if at least one point from $U(x)$ is attacked then $x \in A_{i+1}$.



- On the other hand it may happen as well that all of the points from $\bigcup\{U(x) : x \in K_{i+1}\}$ are covered and still we have $|L_{i+1}| < b_i$. In this case we know that exactly $|L_{i+1}| - \lfloor\sqrt{b_i}\rfloor$ points from B_i were attacked. In particular $|L_{i+1}| - \lfloor\sqrt{b_i}\rfloor - m_i$ points from S_i are attacked so that

$$s_i - |L_{i+1}| + \left\lfloor\sqrt{b_i}\right\rfloor + m_i = b_i + \left\lfloor\sqrt{b_i}\right\rfloor - |L_{i+1}| \geq b_i - |L_{i+1}|$$

gives that there remains at least $b_i - |L_{i+1}|$ unattacked single-colored points (from S_i). Now the Spoiler attacks this number of points and increases the size of the level L_{i+1} to the required size b_i.

After completing i-th round we put

$$B_{i+1} := L_{i+1} - K_{i+1}$$

so that the next base consists of those points from L_{i+1} that cover exactly one point of the base B_i. Consequently we have $b_{i+1} = b_i - \lfloor \sqrt{b_i} \rfloor$ as claimed in (5). We iterate this construction as long as

$$b_i \leq c_i. \tag{15}$$

Now, knowing the construction of the poset we can get back and prove (3). Before showing it let us introduce the following notation, where $i \geq 0$:

- $N_i := \{x \in S_i : x \text{ is not attacked}\}$,
  i.e., N_i consists of non-attacked points from B_i. It should be obvious from what has been already said that

$$|N_i| = \left\lfloor \sqrt{b_i} \right\rfloor. \tag{16}$$

- $S_i' := \{x \in S_i : c(x) \subseteq c(K_{i+1})\}$,
  but we check the coloring before the attack at points from S_i.

- $A_{i+1}' := \{a \in K_{i+1} : \text{all of the points from } U(a) \text{ were attacked}\}$,
  We immediately have $c(A_{i+1}') \cap c(N_i) = \emptyset$. From the definition of set S_i' we get

$$c(A_{i+1}') \cap c(N_i \cap S_i') = \emptyset \tag{17}$$

We are interested in those points that froze colors for further use. In our strategy the Spoiler does not add any point above K_{i+1} and each point from K_{i+1} has to keep at least one color forever. Therefore at least $|K_{i+1}| = \lfloor \sqrt{b_i} \rfloor$ are frozen for further use.

However we can do much better, by showing that each point in $N_i \cup A_{i+1}'$ freezes one color. Since both N_i and A_{i+1}' are antichains, and $c(N_i) \cap c(A_{i+1}') = \emptyset$, we get that at least

$$|N_i| + \left|A_{i+1}'\right|$$

colors are frozen.

We will show that

$$\left|A'_{i+1}\right| \geq \left\lfloor \sqrt{b_i} \right\rfloor - \frac{2b_i}{2b_i - c_i} \tag{18}$$

After we are done with (18) we can argue that at least

$$|N_i| + \left|A'_{i+1}\right| \geq \left\lfloor \sqrt{b_i} \right\rfloor + \left\lfloor \sqrt{b_i} \right\rfloor - \frac{2b_i}{2b_i - c_i} = 2\left\lfloor \sqrt{b_i} \right\rfloor - \frac{2b_i}{2b_i - c_i}$$

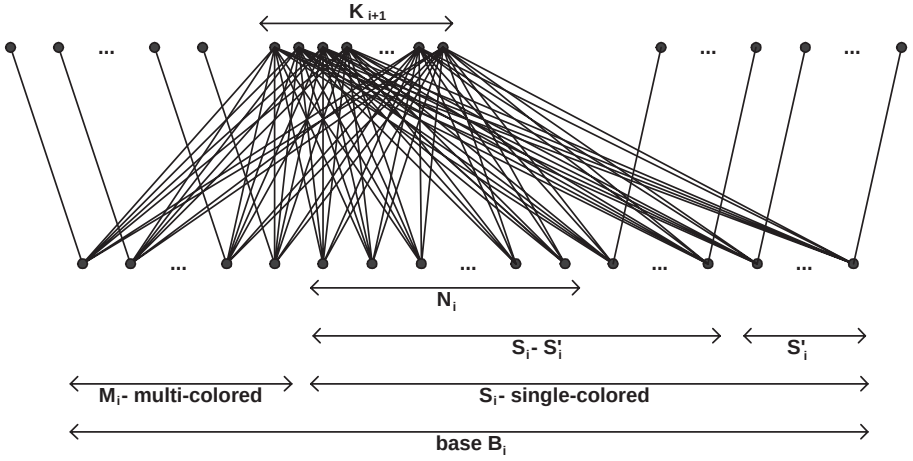
colors are frozen. Thus (2) immediately yields (3) as required. Thus, all we need for (3) is to prove (18).

Before proving this inequality recall that on his way to construct L_{i+1} the Spoiler after attacking M_i and creating K_{i+1} turns to attack some points from S_i. Exploring S_i, first points from S'_i are attacked and he gets to $S_i - S'_i$ only if

$$|S'_i| < |S_i| - \lfloor \sqrt{b_i} \rfloor. \tag{19}$$

Now we split the proof of (18) into two cases.

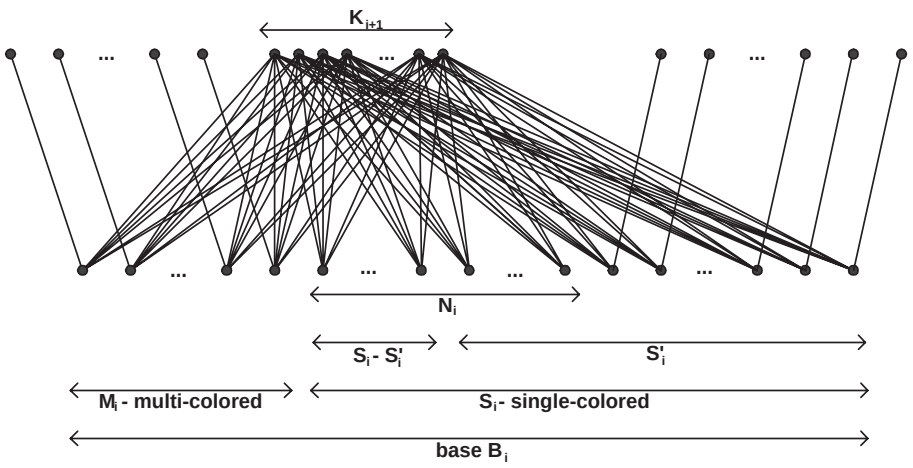
The first one is when (19) holds.



In this case all points from S'_i were attacked so that $S'_i \cap N_i = \emptyset$. Equation (17) immediately gives $A'_{i+1} = K_{i+1}$, making (18) to be trivial.

Our second case is much more laborious. In this setting we have

$$|S_i| - |S'_i| \leq \left\lfloor \sqrt{b_i} \right\rfloor.$$



In this case we know that the construction of L_{i+1} is terminated by the guard (14), so we immediately get

$$S_i - N_i \subseteq U(A_{i+1}). \tag{20}$$

Now we can write

$$|c(A_{i+1}) \cap c(S'_i)| = |U(A_{i+1})| \geq |S_i| - |N_i| = |S_i| - \left\lfloor \sqrt{b_i} \right\rfloor, \tag{21}$$

where again c denotes the coloring before the attack at the point from S_i. First equation is from (13) and the definition of S'_i. The next inequality we get from (20) and the fact that $N_i \subseteq S_i$. The last one we have from (16).

Applying Lemma 10 with elements of K_{i+1} serving as boxes, the colors from $c(S'_i)$ serving as balls that are distributed into the boxes we get

$$|c(A_{i+1}) \cap c(S'_i)| \leq |A_{i+1}| \, \frac{|S'_i|}{\lfloor \sqrt{b_i} \rfloor} \tag{22}$$

Using (10),(21),(22) and the fact that $|S'_i| \leq |S_i|$ we have

$$\begin{aligned} |A_{i+1}| &\geq \frac{|c(A_{i+1}) \cap c(S'_i)| \cdot \lfloor \sqrt{b_i} \rfloor}{|S'_i|} \\ &\geq \frac{\lfloor \sqrt{b_i} \rfloor \cdot (|S_i| - \lfloor \sqrt{b_i} \rfloor)}{|S_i|} \\ &\geq \left\lfloor \sqrt{b_i} \right\rfloor - \frac{b_i}{|S_i|} \\ &\geq \left\lfloor \sqrt{b_i} \right\rfloor - \frac{b_i}{2b_i - c_i} \end{aligned}$$

As we said the guard terminates the construction of L_{i+1} in this case. Let $a \in A_{i+1}$ will be the point last added then from the definition A'_{i+1} we know that for $b \in A_{i+1}, b \neq a$ all points in $U(b)$ are attacked, so we get

$$\text{either} \qquad \left|A'_{i+1}\right| = |A_{i+1}| \qquad \text{or} \qquad \left|A'_{i+1}\right| = |A_{i+1}| - 1.$$

We immediately get

$$\left|A'_{i+1}\right| \geq |A_{i+1}| - 1 \geq \left\lfloor \sqrt{b_i} \right\rfloor - \frac{b_i}{2b_i - c_i} - 1 \geq \left\lfloor \sqrt{b_i} \right\rfloor - \frac{2b_i}{2b_i - c_i}, \qquad (23)$$

where the last inequality immediately follows from (15). This finishes the proof of (18) and therefore (3) is shown as well.

To see that the width of the created poset never exceeds d we induct on q to show that

$$width(\bigcup_{i=0}^{q} L_i) \leq d.$$

This is obvious for L_0, since $width(L_0) = d$. Now for an antichain A in $\bigcup_{i=0}^{q+1} L_i$ decompose the set

$$A^+ := A \cap L_{q+1}$$

into two parts:

$$A^+ \cap K_{q+1}, \qquad A^+ - K_{q+1}$$

Since $|N_i| = \left\lfloor \sqrt{b_i} \right\rfloor = |K_{i+1}|$, we can choose an embedding $g : A^+ \cap K_{i+1} \longrightarrow N_i$. Moreover we can extend g to $f : A^+ \longrightarrow L_q$ by putting $f(x)$ to be the unique subcover of x, whenever $x \in A^+ - K_{q+1}$. Since $f(x) \leq x$ for all $x \in A^+$, the sets

$$f(A^+), \qquad A \cap \bigcup_{i=0}^{q} L_i$$

are disjoint and their join (denoted by A') can be easily seen to be an antichain contained in $\bigcup_{i=0}^{q} L_i$. We can finish the proof by noticing that $|A| = |A'| \leq d$, where the last inequality is nothing else but the induction hypothesis.

Finally we have to show that eventually the Spoiler would force the Algorithm to use strictly more then c colors. This can be enforced on a level L_q on which the Algorithm is left with c_q colors to color an antichain B_q of

size $b_q > c_q$. To see that such a level exists first note that directly from the definition (5) we get that for some p

$$b_0 > b_1 > b_2 > ... > b_p > b_{p+1} = 0.$$

Thus there is q with

$$0 \neq b_q < 2\sqrt{b_0} \tag{24}$$

Now our Claim and (24) yield

$$2b_q - c_q \geq \varepsilon/2 \cdot b_q + 2\sqrt{b_0} \geq 2\sqrt{b_0} > b_q$$

so that the required inequality $b_q > c_q$ holds on q-th level. □

First of all I want to thank Bartlomiej Bosek who is in fact the co-author of this paper. Finally I want to thank our supervisor - prof. Pawel Idziak for motivation as well as Grzegorz Matecki and Kamil Kloch for the help obtained from them.

References

[1] S.Felsner *On-line Chain Partitions of Orders*. Theoretical Computer Science 175:283-292, 1997

[2] H.A.Kierstead, G.F.McNulty and W.T.Trotter. *A theory of recursive dimension for ordered sets*. Order, 1:67-82, 1984.

[3] K.Kloch, *Wyniar off- i on-line częściowych porzadków*, Master Thesis, Computer Science Institute, Jagiellonian University, 2003.