On-line version of Rabinovitch theorem for proper intervals

Bartłomiej Bosek[a], Kamil Kloch[b], Tomasz Krawczyk[a], Piotr Micek[a]

[a] *Algorithmics Research Group, Faculty of Mathematics and Computer Science, Jagiellonian University, Gronostajowa 3, 30-387 Kraków, Poland*
[b] *Embedded Systems Lab, University of Passau, Innstrasse 43, 94032 Passau, Germany*

Abstract

We analyze the on-line dimension of semi-orders as a two-person game between Algorithm and Spoiler, in a customary way. The game is played in rounds. Spoiler presents a collection of intervals representing a semi-order, one interval at a time. Algorithm maintains its realizer, i.e., the set of linear extensions intersecting to the semi-order presented so far. Each time a new interval is presented, Algorithm inserts it into all maintained linear extensions and is not allowed to change the ordering of the previously introduced elements. Reading carefully the theorem of Rabinovitch on dimension of semi-orders one can prove that Algorithm needs only 3 linear extensions when Spoiler presents intervals of unit length. With the introduction of proper intervals, however, Algorithm can be forced to use one more extension. We prove that the value of the game on proper intervals is exactly 4.

Key words: on-line dimension, semi-order

1. Introduction

The concept of dimension of partial orders was introduced more than 60 years ago in Dushnik and Miller's classic paper [1]. Dimension theory has greatly influenced the research on combinatorial properties of posets and graphs. For a comprehensive account on the topic and an extensive bibliography work we refer the reader to Trotter's monograph [2].

When $\mathbf{P} = (X, P)$ and $\mathbf{Q} = (X, Q)$ are partial orders on the same set X then we call $\mathbf{Q}$ an *extension* of $\mathbf{P}$ if $P \subseteq Q$, i.e., if $x \leqslant y$ in $\mathbf{P}$ implies $x \leqslant y$ in $\mathbf{Q}$, for

Email addresses: bartlomiej.bosek@tcs.uj.edu.pl (Bartłomiej Bosek), kamil.kloch@uni-passau.de (Kamil Kloch), tomasz.krawczyk@tcs.uj.edu.pl (Tomasz Krawczyk), piotr.micek@tcs.uj.edu.pl (Piotr Micek)

URL: http://tcs.uj.edu.pl/Bosek (Bartłomiej Bosek), http://www.esl.fim.uni-passau.de/~kloch/ (Kamil Kloch), http://tcs.uj.edu.pl/Krawczyk (Tomasz Krawczyk), http://tcs.uj.edu.pl/Micek (Piotr Micek)

all $x, y \in X$. Among all extensions of a given poset, those which are additionally linear orders are of special importance. They are called *linear extensions*. For a poset $\mathbf{P}$ consisting of n elements $x_1, \ldots, x_n$ we write $L = (x_1, \ldots, x_n)$ as an abbreviation for a linear extension $\mathbf{L} = (X, L)$ of $\mathbf{P}$ in which $x_1 < \cdots < x_n$. A set $\mathcal{R}$ of linear extensions of a poset $\mathbf{P}$ intersecting to $\mathbf{P}$ is called a *realizer* of $\mathbf{P}$. This means that for any two incomparable points x, y in $\mathbf{P}$ there are two linear extensions $L_1, L_2 \in \mathcal{R}$ admitting $x < y$ in L_1 and $x > y$ in L_2. The *dimension* of a poset $\mathbf{P}$, denoted by $\dim(\mathbf{P})$, is the least integer k for which there exists a realizer of $\mathbf{P}$ consisting of k linear extensions.

It is well known that the dimension of $\mathbf{P} = (X, \leqslant)$ with $|X| = n$ does not exceed $\frac{n}{2}$. Recall that for a partially ordered set $\mathbf{P}$ the *width* of $\mathbf{P}$ is the size of the largest antichain in $\mathbf{P}$ and the height of $\mathbf{P}$ is the size of the largest chain in $\mathbf{P}$. By Dilworth's theorem any order of width w can be partitioned into w chains. Another classical Dilworth's theorem says that $\dim(\mathbf{P}) \leqslant \mathrm{width}(\mathbf{P})$.

A poset $\mathbf{P} = (X, \leqslant)$ is called an *interval order* if there is a function I which assigns to each $x \in X$ a closed interval $I(x) = [l_x, r_x]$ of the real line so that $x < y$ in $\mathbf{P}$ if and only if $I(x) < I(y)$, i.e., $r_x < l_y$. The function I is called an *interval representation* of $\mathbf{P}$.

It is not entirely naive to ask whether there are interval orders which have large dimension. It turns out that although interval orders have somewhat one-dimensional nature, their dimension can be arbitrarily high. Surprisingly, according to our knowledge, the complexity of determining the dimension of interval orders is still unknown, in contrast to all orders, where the problem is known to be NP-hard.

An interval order $\mathbf{P} = (X, \leqslant)$ is called a *semi-order* if it has an interval representation $\{[l_x, r_x] : x \in X\}$ such that $r_x = l_x + 1$ for every $x \in X$. By possibly locally stretching some of the intervals one can easily show that $\mathbf{P}$ is a semi-order if it admits a *proper* interval representation, i.e., a representation in which no interval is properly contained in another one.

A poset which admits a partition of its elements into antichains $A_1, \ldots, A_n$ such that $A_1 < A_2 < \ldots < A_n$ (i.e. $a_i < a_j$ for $a_i \in A_i$, $a_j \in A_j$ and $i < j$) is called a *weak order*. It is easy to see that every weak order is also a semi-order.

Theorem 1 (Rabinovitch [3])**.** *The dimension of a semi-order is at most* 3*.*

The bound obtained in Theorem 1 is tight, i.e., semi-orders of dimension 3 do exist. In fact, the original result of Rabinovitch lists four 3-dimensional semi-orders and proves that every semi-order of dimension 3 must contain at least one of these four orders as a subposet.

All mentioned basic parameters of orders: width, height and dimension have their witnessing structures. These structures in are: chain decomposition, antichain decomposition and a realizer, all of the smallest possible size equal to the respective parameter. In the on-line setting the sole question about the value of those parameters is not that interesting, as all of them can be computed after each round of the game exactly in the same way as in the off-line case. Therefore, instead of asking only for the scalar values we additionally require to build

(and update on-line) an appropriate witnessing structure, in our case an on-line realizer.

The on-line dimension of orders is defined as an outcome of a two-person game. We call the players Spoiler and Algorithm. The game is played in rounds. Spoiler presents an on-line order, one point at a time. Algorithm maintains its realizer, i.e., the set of linear extensions intersecting to the order presented so far. It is forbidden for Algorithm to change the ordering of the previously introduced elements in the existing linear extensions. The performance of Algorithm is measured by comparing the number of linear extensions used against the off-line width of the presented poset. The *value* of the game, denoted by $\mathrm{val}(w)$ is the least integer k such that Algorithm has a strategy using at most k chains on any on-line order of width w presented by Spoiler.

For the unrestricted class of all orders Kierstead, McNulty and Trotter [4] proved that $\mathrm{val}(3) = \infty$, i.e. Algorithm can be forced to construct an arbitrarily large realizer already on orders of width 3. In the very same paper they also prove that if Spoiler presents an on-line order of width w without an n-crown as a subposet for any $n \geqslant 3$ then the number of linear extensions needed by Algorithm is indeed bounded in terms of w. This brings up the question whether on-line dimension is perhaps finite on other classes of orders defined in terms of forbidden structures, like interval orders or semi-orders?

The on-line dimension of interval orders is still far from being understood. For example, instead of presenting merely points, Spoiler may reveal the underlying interval representation of the poset. The result of the two games will be completely different. As interval orders do not induce n-crown for $n \geqslant 3$ their on-line dimension is bounded in terms of the width. The results of Hopkins [5], Kierstead, McNulty, Trotter [4] and Felsner [6] give us some better bounds (see Table 1).

presentation method	**bounds**	**remarks**
w/o representation	$\frac{4}{3}w \leqslant ? \leqslant 4w - 4$	[4], [5]
with representation	$? \leqslant \log(w)$	[6]

Table 1: On-line dimension of interval orders

In this paper we investigate the on-line dimension of semi-orders. First, inspired by the proof technique of Rabinovitch theorem, we show that Algorithm can maintain an on-line realizer of size 3 if intervals presented by Spoiler are of the same length. Since semi-orders of dimension 3 do exist, the achieved result is optimal. Next, in Section 3, we deal with the proper interval representation, i.e., the case in which the presented intervals may be arbitrarily long but none of them can contain another one. For this variant we prove a matching lower and upper bound of 4.

presentation method	bounds	remarks
w/o representation	$\frac{4}{3}w \leqslant ? \leqslant 2w$	[4], [7]
proper representation	4	Theorem 3
unit representation	3	Corollary 2

Table 2: On-line dimension of semi orders

2. Proof of Rabinovitch theorem

When $\mathbf{P} = (X, \leqslant)$ is an interval order with a representation I, the relation between the elements of $\mathbf{P}$ can be easily obtained from the set $\{I(x) : x \in X\}$ and so we use the bold symbol $\mathbf{I}$ when considering the poset $\mathbf{P}$ with its interval representation I. An interval representation $\mathbf{I}$ is *distinguishing* if all end points of the intervals are distinct numbers (in particular, no interval is degenerate).

Let $\mathbf{I}$ be an interval representation. An injective function $\mu\colon \mathbf{I} \to \mathbb{R}$ is called a *marking* function on $\mathbf{I}$ if for every interval $x \in \mathbf{I}$ we have $\mu(x) \in x$. Marking function μ naturally defines a linear extension L_μ of $\mathbf{I}$ in which $x < y$ if and only if $\mu(x) < \mu(y)$.

We now give the proof of Rabinovitch theorem. The reason for including the proof of such a classic result shall become apparent later in Sec. 3.2, when proving the upper bound of Theorem 3.

Proof of Theorem 1. Let $\mathbf{P}$ be a semi-order and $\mathbf{I}$ a distinguishing, unit-length representation of $\mathbf{P}$. Any interval $x \in \mathbf{I}$ contains one or two integer points but we can always shift the whole representation in such a way that each interval contains exactly one such point. Let $f : \mathbf{I} \to \mathbb{Z}$ be the function assigning to an interval x its unique integral grid point. Note that:

(i) $f^{-1}(k)$ is an antichain in $\mathbf{I}$,
(ii) $f^{-1}(k) < k + 1 < f^{-1}(k + 2)$.

Define a partition of $\mathbf{I}$ into the disjoint sum $\mathbf{I}_0 \cup \mathbf{I}_1$ as follows:

$$\mathbf{I}_0 \coloneqq f^{-1}(2\,\mathbb{Z}), \quad \mathbf{I}_1 \coloneqq f^{-1}(2\,\mathbb{Z} + 1).$$

From (i) and (ii) we get that both $\mathbf{I}_0$ and $\mathbf{I}_1$ are serial compositions of antichains, i.e., weak orders. Now define the marking functions μ_j, for $j = 0, 1$:

$$\mu_j(x) = \begin{cases} r_x, & \text{if } x \in \mathbf{I}_j, \\ l_x, & \text{otherwise,} \end{cases}$$

for $x = [l_x, r_x] \in \mathbf{I}$. Let $L_0 = L_{\mu_0}$ and $L_1 = L_{\mu_1}$. Define the third linear extension L_2 so that

– if $f(x) < f(y)$ then $x < y$ in L_2,
– if $f(x) = f(y)$ then $x < y$ in L_2 if and only if $x > y$ in L_0.

Linear extension L_2 orders intervals according to their f-value, i.e., their grid point, and in case of equal f-values the ordering is a reverse of L_0. A linear order L_2 is indeed an extension of $\mathbf{I}$ as $x < y$ implies $f(x) < f(y)$.

We claim that $\{L_0, L_1, L_2\}$ is a realizer of $\mathbf{I}$. Choose an incomparable pair $x \parallel y$. There are now two possibilities: $|f(x) - f(y)| = 1$ or $f(x) = f(y)$. In the former case without loss of generality we may assume that $x \in \mathbf{I}_0$, $y \in \mathbf{I}_1$ and $l_x < l_y < r_x < r_y$. Thus $\mu_0(x) > \mu_0(y)$, $\mu_1(x) < \mu_1(y)$ and therefore $x > y$ in L_0 and $x < y$ in L_1 (see Figure 1). In the latter case, $f(x) = f(y)$ implies $x, y \in \mathbf{I}_j$. By definition of L_2, points x and y are sorted in the opposite order in linear extensions L_0 and L_2. □

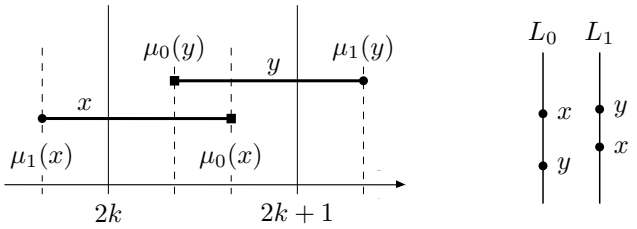

Figure 1: Marking functions for $x \parallel y$ with $x \in \mathbf{I}_0$, $y \in \mathbf{I}_1$

It is almost clear that the realizer $\{L_0, L_1, L_2\}$ from the proof above can be constructed in the on-line setting. Indeed, the partition $\mathbf{I} = \mathbf{I}_0 \cup \mathbf{I}_1$ can be done on-line as incorporation of an incoming interval x into $\mathbf{I}_0$ or $\mathbf{I}_1$ depends only on $f(x)$. Then L_0, L_1 and consequently L_2 depend only on $\mathbf{I}_0$ and $\mathbf{I}_1$. The assumptions that incoming representation is distinguishing and that each interval x contains exactly one integral point $f(x)$ can be omitted by differentiating endpoints with some ε's.

Corollary 2. *The value of the on-line dimension game for a unit-length interval representation is* 3.

3. On-line game on proper intervals

We now consider the case when the semi-order presented by Spoiler is given by a proper interval representation, i.e., a representation in which intervals may be arbitrarily long but none of them may be properly contained in another one. In particular, Algorithm can no longer use a unit-length grid to partition the incoming intervals into two weak orders as in the proof of Rabinovitch theorem. Instead we have the following result:

Theorem 3. *The value of the on-line dimension game for a proper interval representation is* 4.

3.1. Lower bound

Assume that 3 linear extensions L_1, L_2 and L_3 suffice to maintain a realizer of an on-line proper representation of a semi-order. We present a strategy for Spoiler which forces Algorithm to use the 4-th linear extension. The strategy is presented in phases. Already in Phase 1 we make use of the integer value N which will be calculated at the end of the proof so that all needed Ramsey-style arguments could be carried out. For a curious reader we may reveal that $N = 72$.

Phase 1. Spoiler fixes an integer $n \geqslant N$ and presents n intervals $a_1, \ldots, a_n$ forming an antichain, with end points sorted as in Figure 2. For $i = 1, \ldots, n/2$,

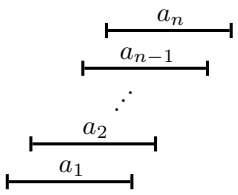

Figure 2: Phase 1

consider $n/2$ pairs of intervals (a_{2i-1}, a_{2i}). Since a_{2i-1} and a_{2i} are incomparable, in at least one linear extension out of L_1, L_2 and L_3, we have $a_{2i-1} > a_{2i}$. Hence there is one extension, say L_1, so that for $n' \geqslant n/6$ such pairs we have $a_{2i-1} > a_{2i}$ in L_1. For the clarity of further consideration we renumber a_i's so that for $i = 1, \ldots, n'$ we have $a_{2i-1} > a_{2i}$ in L_1. All other points can actually be now omitted but they are kept in mind only to prevent Spoiler from presenting a new interval in a way that it contains or is contained in one of them. Actually they can be omitted as all but one intervals presented by Spoiler will be of the same length. The one exceptional interval will arrive at the very end and then we will analyze it carefully.

Phase 2. Spoiler presents n' intervals $b_1, b_3, \ldots, b_{2n'-1}$, again forming an antichain and such that $b_i \parallel a_{i+1}, \ldots, a_{2n'}$ and $b_i > a_i, \ldots, a_1$ (see Figure 3). Recall that $a_{2i-1} > a_{2i}$ in L_1. This, together with $b_{2i-1} > a_{2i-1}$, gives $b_{2i-1} >$

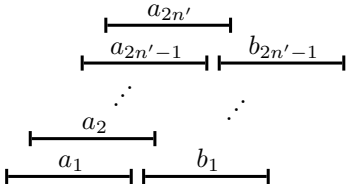

Figure 3: Phase 2

a_{2i} in L_1. Since $b_{2i-1} \parallel a_{2i}$, either in L_2 or L_3 we must have $b_{2i-1} < a_{2i}$. Without loss of generality we may assume that for $n'' \geqslant n'/2$ of the b_i's we have

$b_{2i-1} < a_{2i}$ in L_2. As in Phase 1, we do a renumbering of the important intervals to get $b_{2i-1} < a_{2i}$ in L_2 for $i = 1, \ldots, n''$. After that, we have

$$b_1 < a_2 < b_3 < a_4 < \cdots < a_{2n''-2} < b_{2n''-1} < a_{2n''} \text{ in } L_2. \tag{1}$$

Recall that the k-th *Ramsey* number $\mathrm{R}(k)$ is the smallest integer n so that any graph of order n contains either a k-element clique or a k-element independent set. In the next argument we will merely use the fact that $\mathrm{R}(3) = 6$.

According to (1) we have $b_1 < b_3 < \cdots < b_{2n''-1}$ in L_2. If Algorithm wants to keep a realizer of size 3, then for $1 \leqslant i < j \leqslant n''$ it must be $b_{2i-1} > b_{2j-1}$ either in L_1 or in L_3. If only $n'' \geqslant 6$ then, since $\mathrm{R}(3) = 6$, we find three indices i_0, j_0 and k_0 so that $1 \leqslant i_0 < j_0 < k_0 \leqslant n''$ and $b_{2i_0-1} > b_{2j_0-1} > b_{2k_0-1}$ in L_s, for $s = 1$ or $s = 3$. As in Phases 1 and 2, we renumber the existing intervals so that $i_0 = 1$, $j_0 = 2$, $k_0 = 3$ and hence

$$b_1 > b_3 > b_5 \text{ in } L_s. \tag{2}$$

Phase 3. Spoiler presents two incomparable intervals c_1, c_3 so that $c_1 \parallel b_3, b_5$ and $c_1 > b_1, a_1, \ldots, a_6$ while $c_3 \parallel b_5$ and $c_3 > b_1, b_3, a_1, \ldots, a_6$ (see Figure 4).

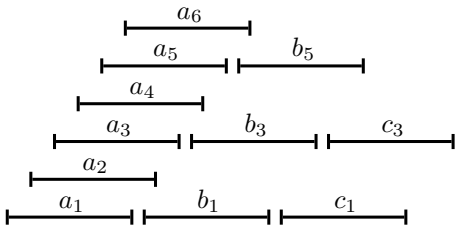

Figure 4: Phase 3

We claim that

$$c_1, c_3 > b_1, b_3, b_5 \text{ both in } L_2 \text{ and } L_s. \tag{3}$$

First, recall from (1) that $a_6 > b_1, b_3, b_5$ in L_2. Since $c_1, c_3 > a_6$, it is clear that $c_1, c_3 > a_6 > b_1, b_3, b_5$ in L_2. Similarly, $c_1, c_3 > b_1, b_3, b_5$ in L_s as $c_1, c_3 > b_1$ and according to (2) we have $b_1 > b_3 > b_5$ in L_s.

Now, since $c_1 \parallel b_3$ and $c_3 \parallel b_5$, from (3) we get $b_3 > c_1$ in L_{4-s} and $b_5 > c_3$ in L_{4-s}. This, together with $c_3 > b_3$, gives

$$b_5 > c_3 > b_3 > c_1 \text{ in } L_{4-s}. \tag{4}$$

Consider what would happen if Spoiler introduced an interval x such that

$$a_1, \ldots, a_6, b_1, b_3 < x \text{ and } x \parallel c_1, c_3, b_5, \tag{5}$$

see Figure 5. From (1) and (2) we get $b_5 < a_6 < x$ in L_2 and $b_5 < b_1 < x$ in L_s, respectively. If Algorithm wants to keep a realizer of size 3, it must put x below b_5 in L_{4-s}.

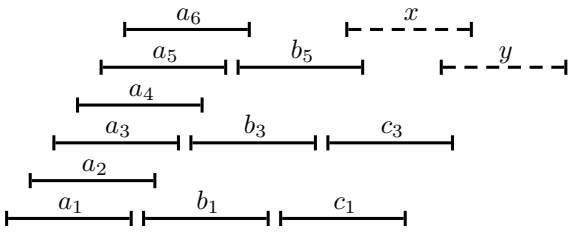

Figure 5: Phase 3—points x and y

Now, consider what would happen if Spoiler introduced y such that

$$a_1, \ldots, a_6, b_1, b_3, b_5, c_1 < y \quad \text{and} \quad y \parallel c_3, \tag{6}$$

see Figure 5. Observe, that (4) together with $c_1 \parallel c_3$ implies $c_1 > c_3$ in L_s or in L_2. Furthermore, $y > b_5 > c_3$ in L_{4-s}. Hence if Algorithm wants to keep a realizer of size 3, there is exactly one linear extension where y can be put below c_3, namely L_s or L_2. We have just proved the following observation.

Observation 4. *Assume that Spoiler plays the strategy described in Phases 1–3 and Algorithm builds a realizer using 3 linear extensions L_1, L_2 and L_3. Then there exist 2 distinct indices $i_0, j_0 \in \{1, 2, 3\}$ such that*

(i) *if Spoiler presents interval x satisfying (5) then $x < b_5$ in L_{i_0},*
(ii) *if Spoiler presents interval y satisfying (6) then $y < c_3$ in L_{j_0}.*

Phase 4. Spoiler plays the mirror-flipped strategies from Phases 1–3 completely to the right, and far apart from the existing intervals, so that the resulting family of intervals looks as in Figure 6. Note that in all 3 linear extensions L_1, L_2 and L_3 we have $b_5, c_1, c_3 < b'_5, c'_1, c'_3$. Now, Observation 4 translated for Phase 4 looks as follows.

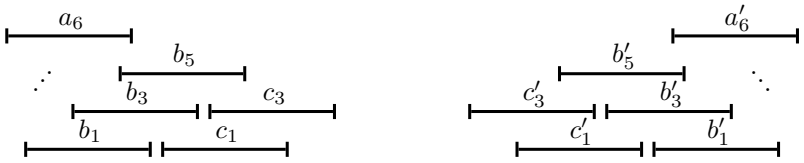

Figure 6: Phase 4

Observation 5. *Assume that Spoiler plays the strategy described in Phases 1–4 and Algorithm builds a realizer using 3 linear extensions L_1, L_2 and L_3. Then there exist 2 distinct indices $i'_0, j'_0 \in \{1, 2, 3\}$ such that*

(i) *if Spoiler presents x' satisfying (5′) that is dual to (5) then $x' > b'_5$ in $L_{i'_0}$,*
(ii) *if Spoiler presents y' satisfying (6′) that is dual to (6) then $y' > c'_3$ in $L_{j'_0}$.*

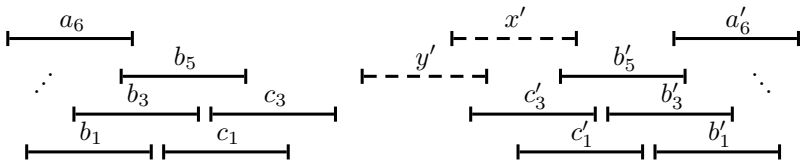

Figure 7: Intervals x' and y' from Observation 5

Phase 5. A careful reader will easily notice that strategies described in Phases 1–4 could be carried out so that all presented intervals are of unit length. It is only in Phase 5 where Spoiler takes advantage of relaxing the unit-length to a proper interval representation of the poset. The indices i_0, j_0, i'_0, j'_0 forced by Observations 4 and 5 obviously satisfy $\{i_0, j_0\} \cap \{i'_0, j'_0\} \neq \emptyset$. The final attack of Spoiler depends on the intersection of those two sets.

(i) If $i_0 = i'_0$ then Spoiler introduces x_1 which plays the role of x to the left part and x' to the right part as in Figure 8. Now Observations 4(i) and 5(i) give $x_1 < b_5$ and $x_1 > b'_5$ in $L_{i_0} = L_{i'_0}$. This is impossible as $b_5 < b'_5$.

(ii) If $i_0 = j'_0$ then Spoiler introduces x_2 which plays the role of x to the left part and y' to the right part as in Figure 8. Now Observations 4(i) and 5(ii) give $x_2 < b_5$ and $x_2 > c'_3$ in $L_{i_0} = L_{j'_0}$. This is impossible as $b_5 < c'_3$.

(iii) If $j_0 = i'_0$ then Spoiler introduces x_3 which plays the role of y to the left part and x' to the right part as in Figure 8. Now Observations 4(ii) and 5(i) give $x_3 < c_3$ and $x_3 > b'_5$ in $L_{j_0} = L_{i'_0}$. This is impossible as $c_3 < b'_5$.

(iv) If $j_0 = j'_0$ then Spoiler introduces x_4 which plays the role of y to the left part and y' to the right part as in Figure 8. Now Observations 4(ii) and 5(ii) give $x_4 < c_3$ and $x_4 > c'_3$ in $L_{j_0} = L_{j'_0}$. This is impossible as $c_3 < c'_3$.

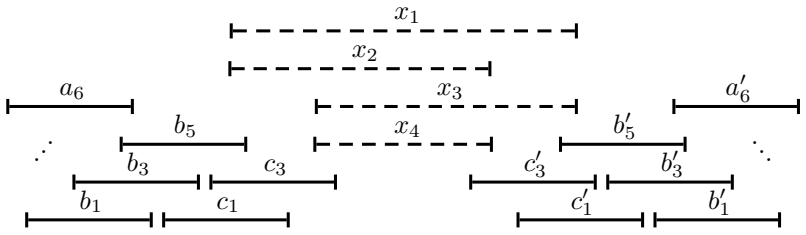

Figure 8: Phase 5—four possible final attacks of Spoiler

To finish the proof we estimate the value of N. The strategy of Spoiler is successful if only $n'' \geqslant \mathrm{R}(3)$. Thus, it suffices to put $n \geqslant N = \mathrm{R}(3) \cdot 2 \cdot 6 = 72$.

3.2. Upper bound

Taking a second look into the proof of Theorem 1 we may notice that it consisted of two independent parts. First, using a unit-length grid, the interval representation was partitioned into 2 weak orders. These weak orders, in turn, defined 3 linear extensions which yielded a realizer. In our current setting the

presented intervals may be arbitrarily long and so a unit-length grid does not induce a partition into 2 weak orders as before. However, if we could find another way of partitioning incoming intervals into (possibly more than two) weak orders, we could follow the second part of the proof of Theorem 1 to obtain the desired realizer. To achieve the first goal we introduce a new game in which Algorithm, instead of a realizer, maintains a partition of incoming intervals into weak orders. Then, adding a twist to the proof technique from Theorem 1, we show that an on-line partition into k weak orders can be transformed into an on-line realizer of size $k+1$.

In order to keep the forthcoming arguments as simple as possible, we additionally assume that the on-line interval representation $\mathbf{I}$ presented by Spoiler is distinguishing, i.e., that no two intervals share the same end point. This condition, although reducing the set of strategies which could have been possibly played by Spoiler, does not change the value k of the game. Indeed, any strategy that would allow Spoiler to force more than k weak orders, would have used only finitely many points. Since any finite interval order admits a distinguishing interval representation, this strategy could be transformed into the one using a distinguishing interval representation.

Let $\mathbf{I}$ be a proper interval representation of a weak order, with its serial decomposition into antichains $\mathbf{I} = A_1 \cup \ldots \cup A_m$ such that $A_i < A_j$, for $i < j$. Obviously the interval $a_i = \bigcap_{x \in A_i} x$ is non-empty, while $a_i \cap a_j = \emptyset$ for $i \neq j$. Let $C(\mathbf{I}) = \{a_1, \ldots, a_m\}$. We say that a_i is the *core* of each $x \in A_i$. Obviously every $x \in A_i$ contains exactly one core, namely the a_i.

Our weak order partitioning game is defined as follows. Spoiler presents an on-line proper interval representation $\mathbf{I}$. Algorithm partitions $\mathbf{I}$ into pairwise disjoint weak orders $\mathbf{I}_1, \ldots, \mathbf{I}_k$ so that cores from the set $\mathbf{C} \coloneqq C(\mathbf{I}_1) \cup \ldots \cup C(\mathbf{I}_k)$ are pairwise disjoint, i.e., the set $\mathbf{C}$ is a linear order. This linear ordering of $\mathbf{C}$ is required to compensate for the lack of a unit-length grid that had produced $\{f^{-1}(i) : i \in \mathbb{Z}\}$ in the proof of Theorem 1.

The least k for which Algorithm has a strategy partitioning any given proper interval representation into k weak orders is called the value of the weak order partitioning game. This value gives the following upper bound for the on-line dimension game.

Proposition 6. *Denote by k the value of the weak order partitioning game described above. Then the value of the on-line dimension game for the proper interval representation is bounded from above by $k+1$.*

Proof. Assume that the proper representation $\mathbf{I}$ is being on-line partitioned into k weak orders $\mathbf{I}_1, \ldots, \mathbf{I}_k$ and that core intervals from the set $\mathbf{C} = C(\mathbf{I}_1) \cup \ldots \cup C(\mathbf{I}_k)$ are pairwise disjoint at any moment during the game. Note that although $x \in \mathbf{I}$ may contain more than one core from $\mathbf{C}$, every $x \in \mathbf{I}_i$ contains exactly one core from $C(\mathbf{I}_i)$. We let the function $\mathrm{core} : \mathbf{I} \to \mathbf{C}$ assign to $x \in \mathbf{I}_i$ this unique core $a \in C(\mathbf{I}_i)$ for which $x \supseteq a$.

Define marking functions μ_i for $i = 1, \ldots, k$ as follows:

$$\mu_i(x) = \begin{cases} r_x, & \text{if } x \in \mathbf{I}_i, \\ l_x, & \text{otherwise,} \end{cases}$$

for $x = [l_x, r_x] \in \mathbf{I}$. Let $L_i = L_{\mu_i}$ for $i = 1, \ldots, k$. This construction assures us that $x > y$ in L_i whenever $x \in \mathbf{I}_i$, $y \notin \mathbf{I}_i$ and $x \parallel y$. One more linear extension L_0 is defined so that

(i) if $\mathrm{core}(x) < \mathrm{core}(y)$ then $x < y$ in L_0,
(ii) if $\mathrm{core}(x) = \mathrm{core}(y)$ then $x < y$ in L_0 if and only if $x > y$ in L_1.

Note that for intervals x and y with $\mathrm{core}(x) < \mathrm{core}(y)$ we trivially have $x \not> y$ as $x \supseteq \mathrm{core}(x)$ and $y \supseteq \mathrm{core}(y)$. This proves that L_0 is indeed a linear extension of $\mathbf{I}$. We claim that $k+1$ linear extensions $L_0, \ldots, L_k$ yield a realizer of $\mathbf{I}$. Choose an incomparable pair $x \parallel y$. If $x \in \mathbf{I}_i$, $y \in \mathbf{I}_j$ and $i \neq j$ then $x > y$ in L_i and $x < y$ in L_j. Otherwise, $x, y \in \mathbf{I}_j$ and since they are incomparable $\mathrm{core}(x) = \mathrm{core}(y)$. Then in the linear extensions L_0, L_1 points x and y are sorted in the opposite order.

Clearly, for a proper distinguishing interval representation $\mathbf{I}$ the functions $\mu_1, \ldots, \mu_k$ (and also the resulting linear extensions $L_1, \ldots, L_k$ determined by these functions) can be constructed on-line. To see that the remaining linear extension L_0 can be built on-line as well, note that the ordering of elements in $\mathbf{C}$ does not change in time as the cores can only shrink during the game. □

Proposition 6 supplies us with a tool which transforms the on-line weak order partition of size k into the on-line realizer of size $k+1$. The next theorem settles the exact value of the weak order partitioning game. The upper bound of 3 translates, by Proposition 6, to an upper bound of 4 for the on-line dimension problem. On the other hand, it can be verified that the achieved lower bound of 3 holds even in the more general case when the cores of the weak orders need not be disjoint. This gives a good illustration of the difference between a unit-length and a proper interval representations, as by Theorem 1 a unit-length representation can be partitioned on-line into 2 weak orders.

Theorem 7. *The value of the on-line weak order partitioning game for the proper interval representation is equal to* 3.

Proof. For the lower bound observe that an on-line partition into 2 weak orders would, by Proposition 6, provide an upper bound of 3 for the on-line dimension game. This, in turn, would contradict the result from Subsection 3.1.

Assume that a proper distinguishing interval representation $\mathbf{I}$ is extended to $\mathbf{I}' = \mathbf{I} \cup \{x\}$. We present an algorithm which assigns the new interval x to one of the 3 existing (possibly empty) weak orders. We do it in three steps. First, we introduce a data structure used by the algorithm. Second, we define a set of invariants which are to be kept during each run of the algorithm. Finally, we present a pseudo-code of the algorithm.

Data structure

First of all, the data structure of Algorithm 1 consists of the on-line proper interval representation $\mathbf{I}$ presented by Spoiler and its partition into three weak orders $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$. The set of cores of $\mathbf{I}_j$ is denoted by $C(\mathbf{I}_j)$. We let $\mathbf{C} = C(\mathbf{I}_1) \cup C(\mathbf{I}_2) \cup C(\mathbf{I}_3)$. The new interval introduced in each round is called x. Algorithm maintains two coloring functions c_l, $\mathrm{c}_r \colon \mathbf{C} \to \{1, 2, 3\}$. Intuitively, function c_l (c_r respectively) colors the left (right) end points of core intervals from $\mathbf{C}$.

Invariants

(I_0) $\mathbf{I}_1 \cup \mathbf{I}_2 \cup \mathbf{I}_3$ yields a partition of $\mathbf{I}$ into 3 weak orders.

(I_1) The set $\mathbf{C}$ of cores is a set of mutually disjoint intervals, i.e., a linear order.

(I_2) For every core $a \in C(\mathbf{I}_i)$ we have $\{i, \mathrm{c}_l(a), \mathrm{c}_r(a)\} = \{1, 2, 3\}$.

(I_3) For every two consecutive cores $a < b$ from $\mathbf{C}$ with $a \in C(\mathbf{I}_i)$, $b \in C(\mathbf{I}_j)$ we have $i \neq j$ and $\mathrm{c}_r(a) \neq \mathrm{c}_l(b)$ (see Figure 9).

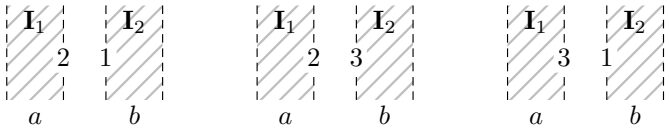

Figure 9: All possible colorings of $\mathrm{c}_r(a)$, $\mathrm{c}_l(b)$, with $a \in C(\mathbf{I}_1)$ and $b \in C(\mathbf{I}_2)$.

(I_4) For $a \in C(\mathbf{I}_i)$ and $y \in \mathbf{I}_j$ with $i \neq j$ we have (see Figure 10)

(i) if $l_a \in y$ then $\mathrm{c}_l(a) = j$,

(ii) if $r_a \in y$ then $\mathrm{c}_r(a) = j$.

Figure 10: Invariant (I_4) shown for $y \in \mathbf{I}_2$ and $a \in C(\mathbf{I}_3)$

Invariant (I_1) guarantees a property relating all three weak orders $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$. It states that the cores of $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$ are mutually disjoint and as such form a linear order. Note that this is nothing else but the rules of the weak order partitioning game. From (I_3) we know that the neighboring cores in the linear order originate from distinct weak orders, and moreover, that the colors of the neighboring end points of the cores must be distinct. Together with (I_2) this induces a very restricted sequence of colors associated with consecutive cores. Finally, for any core $a \in \mathbf{C}$, the value of $\mathrm{c}_l(a)$ ($\mathrm{c}_r(a)$, respectively) determines, by (I_4), the weak order of all intervals which do not contain a but do intersect a.

As in the proof of Proposition 6 we let the function core : $\mathbf{I} \to \mathbf{C}$ assign to $x \in \mathbf{I}_i$ the unique core $a \in C(\mathbf{I}_i)$ for which $x \supseteq a$. Recall that the weak order $\mathbf{I}_i$ is a serial composition of antichains, say $A_1, \ldots, A_m$. If two intervals $x, y \in \mathbf{I}_i$ intersect then x, y are in the same antichain A_j and therefore $\mathrm{core}(x) = \bigcap_{z \in A_j} z = \mathrm{core}(y)$. Hence for two intervals $x, y \in \mathbf{I}_i$ we have

$$x \text{ and } y \text{ intersect if and only if } \mathrm{core}(x) = \mathrm{core}(y). \tag{7}$$

Note also that for $a \in C(\mathbf{I}_i)$ there must exist $y, z \in \mathbf{I}_i$ with $l_y = l_a$, $r_z = r_a$ and $\mathrm{core}(y) = \mathrm{core}(z) = a$ (see Figure 11).

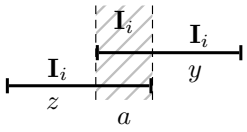

Figure 11: Intervals witnessing the end points of the core

Before presenting our Algorithm we need the following Claim which will help us to split its work into cases.

Claim 8. *Suppose that the data structure* ($\mathbf{I}$, $\mathbf{I}_1$, $\mathbf{I}_2$, $\mathbf{I}_3$, c_l, c_r) *satisfies properties described by* (I_0)–(I_4). *Then*

(i) *for every $y \in \mathbf{I}$ there is exactly one $a \in \mathbf{C}$ such that $a \subseteq y$,*
(ii) *if $\mathbf{I}$ is extended to $\mathbf{I}' = \mathbf{I} \cup \{x\}$ then the new interval x contains at most one core interval from $\mathbf{C}$.*

Proof. Suppose that (i) fails, i.e., there is $y \in \mathbf{I}_k$ which contains two core intervals $a < b$. By (I_1) we may assume that a and b are consecutive in $\mathbf{C}$. From (I_3) it now follows that a and b originate from two different weak orders, i.e., that $a \in C(\mathbf{I}_i)$, $b \in C(\mathbf{I}_j)$ and $i \neq j$. Without loss of generality we may assume that $y \notin \mathbf{I}_j$, i.e., $j \neq k$. Since b is contained in y with its both ends, namely l_b and r_b, our invariant (I_4) gives $\mathrm{c}_l(b) = k = \mathrm{c}_r(b)$, which contradicts ($\mathrm{I}_2$).

Now, suppose that (ii) fails, i.e., the new interval x contains two core intervals $a < b$, in particular, $l_x < l_a < r_b < r_x$. Like in the proof of (i), by (I_1) we may assume that a and b are consecutive in $\mathbf{C}$, and again from (I_3) we get that $a \in C(\mathbf{I}_i)$, $b \in C(\mathbf{I}_j)$ and $i \neq j$. Since a is the core of $\mathbf{I}_i$ there must be some $y \in \mathbf{I}_i$ witnessing the left end point of a, i.e., such that $l_y = l_a$. Now, from (i) applied to y it follows that $b \not\subseteq y$. Hence $r_y < r_b < r_x$ and so $y \subsetneq x$. This contradicts the fact that $\mathbf{I} \cup \{x\}$ is a proper interval representation. □

Algorithm

Algorithm 1 puts the new interval x into one of the three maintained weak orders $\mathbf{I}_1$, $\mathbf{I}_2$ or $\mathbf{I}_3$ and updates the coloring functions c_l, c_r in such a way that invariants (I_0)–(I_4) are kept. We distinguish the variables before and after the incorporation of x into $\mathbf{I}$ by appending $'$ to the latter ones, i.e., $\mathbf{I}_1$ becomes $\mathbf{I}'_1$, c_r becomes c'_r etc. In particular, $\mathbf{C}' = C(\mathbf{I}'_1) \cup C(\mathbf{I}'_2) \cup C(\mathbf{I}'_3)$. By writing 'put x

into $\mathbf{I}_i$' we mean $\mathbf{I}'_i \leftarrow \mathbf{I}_i \cup \{x\}$ and $\mathbf{I}'_j \leftarrow \mathbf{I}_j$ for $j \neq i$. To simplify Algorithm we consider the initial situation in which Spoiler had already introduced the first interval x_0. For this interval we manually define $\mathbf{I}_1 = \{x_0\}$, $\mathbf{I}_2 = \mathbf{I}_3 = \emptyset$ so that $\mathbf{C} = C(\mathbf{I}_1) = \{x_0\}$ and then we put $\mathrm{c}_l(x_0) = 2$ and $\mathrm{c}_r(x_0) = 3$.

Algorithm 1: Weak order partition of a proper interval representation

```
1  if x ⊇ a for some core a ∈ C(I_i) then                          /* 1 */
2  |  put x into I_i;
3  else if x intersects cores C(I_i) ∋ a < b ∈ C(I_j) then         /* 2 */
4  |  if c_r(a) = j then put x into I_j;
5  |  else if c_l(b) = i then put x into I_i;
6  else if x intersects exactly one core, say a ∈ C(I_i) then     /* 3 */
7  |  put x into I_i;
8  else if x does not intersect any existing core then
9  |  if x lies between consecutive C(I_i) ∋ a < b ∈ C(I_j) then /* 4.1 */
10 |  |  c'_l(x) ← i;
11 |  |  c'_r(x) ← j;
12 |  else if x > a := max(C) with a ∈ C(I_i) then               /* 4.2 */
13 |  |  c'_l(x) ← i;
14 |  |  c'_r(x) ← color distinct from i;
15 |  else if x < b := min(C) with b ∈ C(I_j) then               /* 4.3 */
16 |  |  c'_r(x) ← j;
17 |  |  c'_l(x) ← color distinct from j;
18 |  put x into I_m so that {m, c'_l(x), c'_r(x)} = {1, 2, 3}
19 foreach a' ∈ C' such that a' ⊆ a for some a ∈ C do
20 |  c'_l(a') ← c_l(a);
21 |  c'_r(a') ← c_r(a);
```

As it can be easily seen, the weak order that incorporates the incoming x depends on how x interacts with the existing cores. Due to Claim 8(ii) this behavior is covered by six cases 1–3 and 4.1–4.3. All we need to show is that our extended data structure satisfies (I_0)–(I_4).

Case 1

In this case we have $x \supseteq a$ for some core interval $a \in C(\mathbf{I}_i)$. By Claim 8(ii) we know that a is the unique core interval which is contained in x.

In order to prove (I_0) we show that $\mathbf{I}'_i := \mathbf{I}_i \cup \{x\}$ is a weak order. Clearly, the set $\{x\} \cup \mathrm{core}^{-1}(a)$ is an antichain as x intersects every $y \in \mathbf{I}_i$ for which $\mathrm{core}(y) = a$. It remains to show that x is disjoint with all other intervals from $\mathbf{I}_i$. Suppose to the contrary that for some $y \in \mathbf{I}_i$ with $\mathrm{core}(y) = b \neq a$ we have $x \cap y \neq \emptyset$. Without loss of generality assume that $a < b$ and so $r_x \in y$ (see Figure 12). Let $z \in \mathbf{I}_i$ be an interval witnessing the left end point of the core a, i.e., $l_z = l_a$. Since $\mathbf{I} \cup \{x\}$ is a distinguishing proper interval representation and $a \subseteq x$, we must have $l_x < l_z$. Hence $r_x < r_z$, as otherwise z would be properly

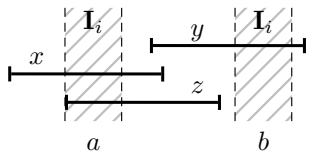

Figure 12: Case 1, proof of (I_0)

contained in x. The latter implies $z \cap y \neq \emptyset$. This, in turn, contradicts (7).

Invariants (I_1)–(I_3) are trivially satisfied as the set $\mathbf{C}$ of cores and the coloring functions c_l, c_r remain unchanged. However, some effort is needed to prove that the last invariant (I_4) is kept.

Suppose that x intersects the core $b \in C(\mathbf{I}_j)$ and $j \neq i$. Since a is the unique core interval which is contained in x, we get $b \nsubseteq x$. Suppose that $a < b$. Then b must be the immediate successor of a in $\mathbf{C}$. To prove (I_4) for x and b we need to show that $\mathrm{c}_l(b) = i$. Let $z \in \mathbf{I}_i$ be an interval witnessing the left end point of a, i.e., $l_z = l_a$ (see Figure 13). Now, since $\mathbf{I} \cup \{x\}$ is a distinguishing proper

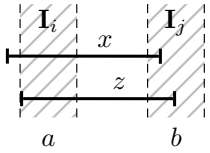

Figure 13: Case 1, proof of (I_4)

interval representation, we have $l_x < l_z = l_a$ and therefore $r_x < r_z$. Hence $l_b \in z$. Invariant (I_4) applied to b and z gives $\mathrm{c}_l(b) = i$, as desired. An analogous proof delivers $\mathrm{c}_r(b) = i$ in the case when $b < a$.

Case 2

In this case the new interval x intersects exactly two consecutive core intervals $a < b$ with $a \in C(\mathbf{I}_i)$ and $b \in C(\mathbf{I}_j)$. By (I_3) we know that $i \neq j$. Since we are not in Case 1, we also know that neither a nor b is contained in x. In particular, $l_a < l_x < r_a < l_b < r_x < r_b$.

First of all, we need to show that lines 4–5 of Algorithm 1 cover all possibilities, i.e., either $\mathrm{c}_r(a) = j$ or $\mathrm{c}_l(b) = i$. Suppose that $\mathrm{c}_r(a) \neq j$. Invariant (I_2) applied to $a \in C(\mathbf{I}_i)$ implies $\mathrm{c}_r(a) \neq i$. Hence $\{i, j, \mathrm{c}_r(a)\} = \{1, 2, 3\}$. But this restricts the possible values of $\mathrm{c}_l(b)$. Indeed, (I_2) applied to $b \in C(\mathbf{I}_j)$ together with (I_3) give $\mathrm{c}_l(b) \neq j$ and $\mathrm{c}_l(b) \neq \mathrm{c}_r(a)$, respectively. Hence we must have $\mathrm{c}_l(b) = i$, exactly as stated in line 5.

To prove that (I_0)–(I_4) are kept we assume that $\mathrm{c}_r(a) = j$, i.e., line 4 of Algorithm 1 rather than line 5 is executed. The arguments in the case $\mathrm{c}_l(b) = i$ are analogous. For (I_0) we show that $\mathbf{I}'_j \coloneqq \mathbf{I}_j \cup \{x\}$ is a weak order. Clearly, the set $\{x\} \cup \mathrm{core}^{-1}(b)$ is an antichain. It remains to show that x is disjoint with all other intervals from $\mathbf{I}_j$. Suppose to the contrary that for some $y \in \mathbf{I}_j$ with $\mathrm{core}(y) \neq b$ we have $x \cap y \neq \emptyset$. As a and b are consecutive in $\mathbf{C}$, either

core$(y) < a$ or core$(y) > b$. In the latter case $x \cap y \neq \emptyset$ together with $r_x < r_b$ implies $y \cap b \neq \emptyset$ (see Figure 14). Therefore, $y \cap z \neq \emptyset$ for every z with core$(z) = b$,

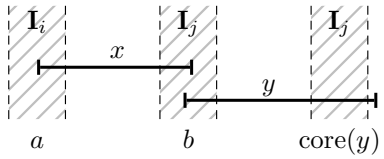

Figure 14: Case 2, proof of (I_0), core$(y) > b$

contradicting (7). Now consider the case when core$(y) < a$. Then $x \cap y \neq \emptyset$ together with $l_a < l_x$ gives $l_a \in y$ (see Figure 15). Invariant (I_4) applied to a and

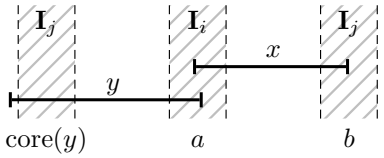

Figure 15: Case 2, proof of (I_0), core$(y) < a$

y gives $\mathrm{c}_l(a) = j$. Hence $\mathrm{c}_l(a) = \mathrm{c}_r(a)$, contradicting ($\mathrm{I}_2$).

Invariants (I_1)–(I_3) are trivially satisfied. Invariant (I_4) holds for x and a as $\mathrm{c}_r(a) = j$.

Case 3

In this case the new interval x intersects, yet does not contain, exactly one $a \in \mathbf{C}$. Without loss of generality assume that $l_a < l_x$. Suppose that (I_0) does not hold, i.e., for some $y \in \mathbf{I}_i$ with core$(y) = b \neq a$ we have $x \cap y \neq \emptyset$. Note that we must have $y \cap a = \emptyset$ as otherwise $y \cap z \neq \emptyset$ for every z with core$(z) = a$, contradicting (7). Together with $l_a < l_x$ this gives $a < y$. Since $a, b \in C(\mathbf{I}_i)$, invariant (I_3) yields a $c \in C(\mathbf{I}_j)$ with $i \neq j$ and $a < c < b$. Interval x intersects only one $a \in \mathbf{C}$. Hence we must have $x < c$ and so $c \subseteq y$ (see Figure 16). But this would mean that y contains two cores b and c, which is impossible, by Claim 8(i).

We easily check that the remaining invariants (I_1)–(I_4) are trivially kept.

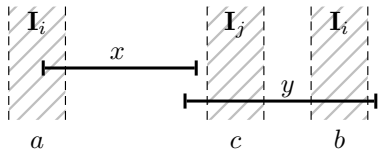

Figure 16: Case 3

Case 4

In this case the new interval x does not intersect any element from $\mathbf{C}$. It is also the only moment when Algorithm forms a brand new core, i.e., $\mathbf{C}' = \mathbf{C} \cup \{x\}$. We only prove that invariants (I_0)–(I_4) are kept in Case 4.1 as the proofs in Cases 4.2 and 4.3 are analogous. In Case 4.1 the new interval x lies between two consecutive core intervals a and b. Thus, intervals $a < x < b$ become three consecutive cores in $\mathbf{C}'$.

In order to prove (I_0) we will show that x does not intersect any interval from $\mathbf{I}_m$. Suppose to the contrary that for some $y \in \mathbf{I}_m$ we have $x \cap y \neq \emptyset$. Without loss of generality assume that $l_x < l_y$. As a and b are consecutive in $\mathbf{C}$, from $y \in \mathbf{I}_m$ and $m \neq j$ we get $b < \mathrm{core}(y)$. Moreover, $l_y < r_x < l_b < r_b < r_y$ (see Figure 17). Thus y contains two cores b and $\mathrm{core}(y)$, which is impossible, by Claim 8(i).

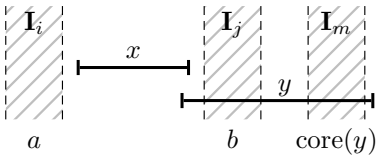

Figure 17: Case 4.1

Invariant (I_1) is trivially satisfied. Invariant (I_3) applied to consecutive (in $\mathbf{I}$) core intervals $a \in C(\mathbf{I}_i)$ and $b \in C(\mathbf{I}_j)$ gives $i \neq j$. Now, $\mathrm{c}'_l(x) = i \neq j = \mathrm{c}'_r(x)$ together with line 18 yields $\{m, \mathrm{c}'_l(x), \mathrm{c}'_r(x)\} = \{1, 2, 3\}$, proving ($\mathrm{I}_2$).

The first part of (I_3) follows again from $\{m, i, j\} = \{1, 2, 3\}$. To verify the second part of (I_3) we need to check that $\mathrm{c}_r(a) \neq \mathrm{c}'_l(x) = i$ and $\mathrm{c}_l(b) \neq \mathrm{c}'_r(x) = j$. Applying ($\mathrm{I}_2$) to a and b we get $\mathrm{c}_r(a) \neq i$ and $\mathrm{c}_l(b) \neq j$, respectively.

To prove (I_4) assume that for some $y \notin \mathbf{I}_m$ we have $x \cap y \neq \emptyset$. If $l_x \in y$ then $\mathrm{core}(y) \neq a$ would imply $\mathrm{core}(y) < a$, $a \subseteq y$ so that y would contain two cores $\mathrm{core}(y)$ and a, which is impossible, by Claim 8(i). Hence $\mathrm{core}(y) = a$, $y \in \mathbf{I}_i$ and so $\mathrm{c}'_l(x) = i$, as desired. The proof in the case $r_x \in y$ is analogous. □

References

[1] B. Dushnik, E. W. Miller, Partially ordered sets, Amer. J. Math. 63 (1941) 600–610.

[2] W. T. Trotter, Combinatorics and partially ordered sets, Johns Hopkins Series in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1992, dimension theory.

[3] I. Rabinovitch, The dimension of semiorders, J. Combin. Theory Ser. A 25 (1) (1978) 50–61.

[4] H. A. Kierstead, G. F. McNulty, W. T. Trotter, Jr., A theory of recursive dimension for ordered sets, Order 1 (1) (1984) 67–82.

[5] L. Hopkins, Some problems involving combinatorial structures determined by intersections of intervals and arcs, Ph.D. thesis, University of South Carolina (1981).

[6] S. Felsner, Interval orders: Combinatorial Structure and Algorithms, Ph.D. thesis, Technische Universität Berlin, http://www.math.tu-berlin.de/~felsner/Paper/diss.pdf (1993).

[7] B. Bosek, K. Kloch, T. Krawczyk, P. Micek, On-line dimension of semi-orders, manuscript (2011).