Lower Bounds on the On-line Chain Partitioning of Semi-orders with Representation

Csaba Biró and Israel R. Curbelo

Rafał Kilar

Thursday 19$^{\text{th}}$ May, 2022

Chain Partitioning

Dilworth's theorem

A poset of width ω can be partitioned off-line into ω chains.

Chain Partitioning

Dilworth's theorem

A poset of width ω can be partitioned off-line into ω chains.

On-line chain partitioning

An on-line chain partitioning algorithm is presented with a poset (X, P) in set order of elements $x_1, x_2, \ldots, x_n$ and constructs an on-line chain partitioning.

On line width

Definition

On-line width $olw(\omega)$ of the class of posets with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any on-line chain partitioning algorithm to use k chains.

On line width

Definition

On-line width $olw(\omega)$ of the class of posets with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any on-line chain partitioning algorithm to use k chains.

The exact value of $olw(\omega)$ is unknown for $\omega > 2$.

Bounds on on-line width

Kierstead 1981

An on-line algorithm that uses at most $(5^\omega - 1)/4$ chains to partition a poset of width ω.

Bounds on on-line width

Kierstead 1981

An on-line algorithm that uses at most $(5^{\omega} - 1)/4$ chains to partition a poset of width ω.

Bosek and Krawczyk 2021

An on-line algorithm that needs at most $\omega^{O(log\ log\ \omega)}$ chains to partition a poset.

Bounds on on-line width

Kierstead 1981

An on-line algorithm that uses at most $(5^{\omega} - 1)/4$ chains to partition a poset of width ω.

Bosek and Krawczyk 2021

An on-line algorithm that needs at most $\omega^{O(log\ \ log\ \omega)}$ chains to partition a poset.

Szemerédi

Any algorithm could be forced to use $\binom{\omega+1}{2}$ chains to partition a poset of width ω.

Bounds on on-line width

Kierstead 1981

An on-line algorithm that uses at most $(5^\omega - 1)/4$ chains to partition a poset of width ω.

Bosek and Krawczyk 2021

An on-line algorithm that needs at most $\omega^{O(\log\ \log\ \omega)}$ chains to partition a poset.

Szemerédi

Any algorithm could be forced to use $\binom{\omega+1}{2}$ chains to partition a poset of width ω.

Bosek, Felsner, Kloch, Krawczyk, Matecki and Micek 2012

Improved lower bound of $(2 - o(1))\binom{\omega+1}{2}$ chains.

Interval Order

Interval order

We call a poset (X, P) an interval order if there exists a funciont I which maps each element $x \in X$ to a closed real number interval $I(x) = [l_x, r_x]$, such that for every $x_1, x_2 \in X$ it holds that $x_1 < x_2$ iff $r_{x_1} < l_{x_2}$.
We call I an interval representation of (X, P).

Interval Order

Interval order

We call a poset (X, P) an interval order if there exists a funciont I which maps each element $x \in X$ to a closed real number interval $I(x) = [l_x, r_x]$, such that for every $x_1, x_2 \in X$ it holds that $x_1 < x_2$ iff $r_{x_1} < l_{x_2}$.
We call I an interval representation of (X, P).

Semi-order

An interval order (X, P) is a semi-order if there is an interval representation I of (X, P) with unit-length intervals $[r_x - 1, r_x]$ on the real line. For each $x_1, x_2 \in X$, $x_1 < x_2$ if and only if $r_{x_1} < r_{x_2} - 1$.

On-Line Width of Interval Orders

On-Line Width of Interval Orders

On-line width $olwi(\omega)$ of the class of interval orders with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any chain partitioning algorithm to use k chains.
Kierstead and Trotter showed that $olwi(\omega) = 3\omega - 2$.

On-Line Width of Interval Orders

On-Line Width of Interval Orders

On-line width $olwi(\omega)$ of the class of interval orders with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any chain partitioning algorithm to use k chains.
Kierstead and Trotter showed that $olwi(\omega) = 3\omega - 2$.

On-Line Width of Interval Orders with Representation

On-line width $olwi_R(\omega)$ of the class of interval orders with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any algorithm to use k chains to partition an interval order of witdth ω presented as intervals.
Instead of presenting the poset as points, it's presented as intervals, which provide an interval representation of a specific poset (X, P). Showed by Chrobak and Ślusarek to be $olwi_R(\omega) = 3\omega - 2$.

On-Line Width of Semi-Orders

On-Line Width of Semi-Orders

On-line width $olws(\omega)$ of the class of semi-orders with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any chain partitioning algorithm to use k chains.
First shown that first-fit algorithm uses $2\omega - 1$ chains. Later it was also proved that any on-line algorithm can be forced to use $2\omega - 1$ chains, giving the exact value $olws(\omega) = 2\omega - 1$

On-Line Width of Semi-Orders

On-Line Width of Semi-Orders

On-line width $olws(\omega)$ of the class of semi-orders with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any chain partitioning algorithm to use k chains.
First shown that first-fit algorithm uses $2\omega - 1$ chains. Later it was also proved that any on-line algorithm can be forced to use $2\omega - 1$ chains, giving the exact value $olws(\omega) = 2\omega - 1$

On-Line Width of Semi-Orders with Representation

On-line width $olws_R(\omega)$ of the class of interval orders with width $\leq \omega$ is the largest $k \in \mathbb{N}$, such that there exists a strategy that forces any algorithm to use k chains to partition when presented with a poset represented with unit-length intervals.
Exact value of $olws_R(\omega)$ is unknown.

On-Line Width of Semi-Orders with Representation

The problem first considered by Chrobak and Ślusarek. As previously mentioned, they showed that first-fit uses at most $2\omega - 1$ chains to perform an on-line chain partitioning. They also showed that any greedy algorithm can be forced to use $2\omega - 1$ chains.

On-Line Width of Semi-Orders with Representation

The problem first considered by Chrobak and Ślusarek. As previously mentioned, they showed that first-fit uses at most $2\omega - 1$ chains to perform an on-line chain partitioning. They also showed that any greedy algorithm can be forced to use $2\omega - 1$ chains.
The question remained, whether more optimal algorithms exists.

On-Line Width of Semi-Orders with Representation

The problem first considered by Chrobak and Ślusarek. As previously mentioned, they showed that first-fit uses at most $2\omega - 1$ chains to perform an on-line chain partitioning. They also showed that any greedy algorithm can be forced to use $2\omega - 1$ chains.
The question remained, whether more optimal algorithms exists.
In 2005, Epstein and Levy showed a strategy, which for any positive integer k forces on-line algorithms to use $3k$ chains to partition a semi-order of width $2k$ represented with intervals.

On-Line Width of Semi-Orders with Representation

The problem first considered by Chrobak and Ślusarek. As previously mentioned, they showed that first-fit uses at most $2\omega - 1$ chains to perform an on-line chain partitioning. They also showed that any greedy algorithm can be forced to use $2\omega - 1$ chains.
The question remained, whether more optimal algorithms exists.
In 2005, Epstein and Levy showed a strategy, which for any positive integer k forces on-line algorithms to use $3k$ chains to partition a semi-order of width $2k$ represented with intervals.
This given us the best bounds known so far:

$$\lfloor \frac{3}{2}\omega \rfloor \leq olws_R(\omega) \leq 2\omega - 1$$

Theorem

We will show a slightly improved lower bound for on-line width of semi-orders with representation.

$$olws_R(\omega) \geq \lceil \frac{3}{2}\omega \rceil$$

Theorem

We will show a slightly improved lower bound for on-line width of semi-orders with representation.

$$olws_R(\omega) \geq \lceil \frac{3}{2}\omega \rceil$$

For a given k we will force any on-line chain partitioning algorithm to use $3k + 2$ chains for a poset of width $\omega = 2k + 1$.

Stage 1

We start with k identical intervals $x_1, \ldots, x_k$ with $x_i = 0$ for each $i \in \{1, \ldots, k\}$.

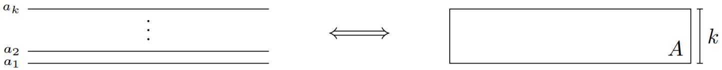
Stage 1

We start with k identical intervals $x_1, \ldots, x_k$ with $x_i = 0$ for each $i \in \{1, \ldots, k\}$.
It's easy to see that they form an antichain, so each of them has to be assigned to a separate chain. Lets denote the set of those chain as $A = \{a_1, \ldots, a_k\}$.

Stage 1

We start with k identical intervals $x_1, \ldots, x_k$ with $x_i = 0$ for each $i \in \{1, \ldots, k\}$.

It's easy to see that they form an antichain, so each of them has to be assigned to a separate chain. Lets denote the set of those chain as $A = \{a_1, \ldots, a_k\}$.



Stage 2

We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$

Stage 2

We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$
2. We present the new interval $x_i = \frac{l_2 + h_2}{2}$

Stage 2

We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$
2. We present the new interval $x_i = \frac{l_2+h_2}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.

Stage 2

We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$
2. We present the new interval $x_i = \frac{l_2 + h_2}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \in A$, then set $h_2 = x_i$

Stage 2

We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$
2. We present the new interval $x_i = \frac{l_2+h_2}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \in A$, then set $h_2 = x_i$
5. If $j \notin A$, then set $l_2 = x_i$

Stage 2

We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$
2. We present the new interval $x_i = \frac{l_2 + h_2}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \in A$, then set $h_2 = x_i$
5. If $j \notin A$, then set $l_2 = x_i$

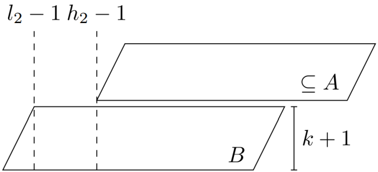
Let B be the set of new chains used by the algorithm. We continue steps 2-5 until $|B| = k + 1$.

Stage 2

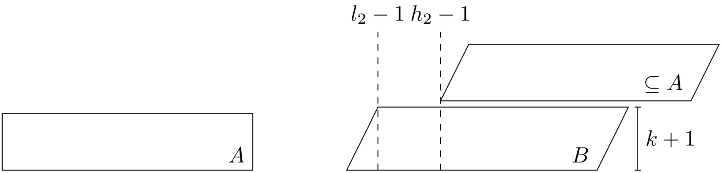
We perform the following steps:

1. We start with $l_2 := 1$ and $h_2 := 2$
2. We present the new interval $x_i = \frac{l_2+h_2}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \in A$, then set $h_2 = x_i$
5. If $j \notin A$, then set $l_2 = x_i$

Let B be the set of new chains used by the algorithm. We continue steps 2-5 until $|B| = k + 1$.



Stage 2



Because $1 < x_i < 2$, all the intervals created in this stage form an antichain of size at most ω.
All of them have to be assigned to different chains. At most k are in A and exactly $k + 1$ in B.

Stage 3

We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$

Stage 3

We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$
2. We present the new interval $x_i = \frac{l_3 + h_3}{2}$

Stage 3

We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$
2. We present the new interval $x_i = \frac{l_3 + h_3}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.

Stage 3

We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$
2. We present the new interval $x_i = \frac{l_3 + h_3}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \notin B$, then set $l_3 = x_i$ and go to round $i + 1$

Stage 3

We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$
2. We present the new interval $x_i = \frac{l_3+h_3}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \notin B$, then set $l_3 = x_i$ and go to round $i + 1$
5. If $j \in B$, then set $h_3 = x_i$ and go to next stage

Stage 3

We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$
2. We present the new interval $x_i = \frac{l_3 + h_3}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \notin B$, then set $l_3 = x_i$ and go to round $i + 1$
5. If $j \in B$, then set $h_3 = x_i$ and go to next stage

For all new intervals x_i in this stage we have $-2 < x_i < -1$, so they form an antichain of size at most ω.

Stage 3

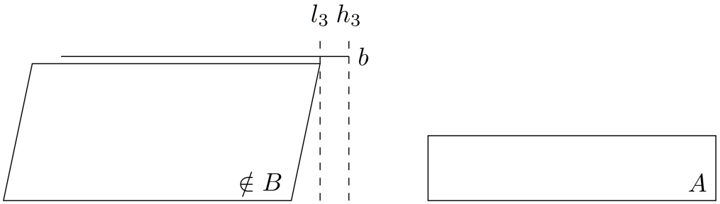
We perform the following steps:

1. We start with $l_3 := l_2 - 3$ and $h_3 := h_2 - 3$
2. We present the new interval $x_i = \frac{l_3+h_3}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
4. If $j \notin B$, then set $l_3 = x_i$ and go to round $i+1$
5. If $j \in B$, then set $h_3 = x_i$ and go to next stage

For all new intervals x_i in this stage we have $-2 < x_i < -1$, so they form an antichain of size at most ω.

If the intervals are assigned to $k+1$ new chains, then we can finish.

We can assume that we finish the stage with an interval $x_B = h_3$ assigned to a chain $b \in B$



Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$

Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$

Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
   We know that $-1 < x_i < x_B + 1 < 0$, so $j \notin A$ and $j \neq b$

Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
   We know that $-1 < x_i < x_B + 1 < 0$, so $j \notin A$ and $j \neq b$
4. Update $l_4 = x_i$

Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$
3. Let j be the chain x_i was assigned to by the algorithm. We know that $-1 < x_i < x_B + 1 < 0$, so $j \notin A$ and $j \neq b$
4. Update $l_4 = x_i$
5. If $j \in B$, the we go to round $i + 1$

Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
   We know that $-1 < x_i < x_B + 1 < 0$, so $j \notin A$ and $j \neq b$
4. Update $l_4 = x_i$
5. If $j \in B$, the we go to round $i + 1$
6. If $j \notin B$, then we move to the next stage

Stage 4

We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$
3. Let j be the chain x_i was assigned to by the algorithm. We know that $-1 < x_i < x_B + 1 < 0$, so $j \notin A$ and $j \neq b$
4. Update $l_4 = x_i$
5. If $j \in B$, the we go to round $i + 1$
6. If $j \notin B$, then we move to the next stage

The new intervals form an antichain of size at most $k + 1$. They all had to be assigned to different chains.
None of the those chains are in $A \cup \{b\}$.
At most k of them are in $B \setminus \{b\}$

Stage 4

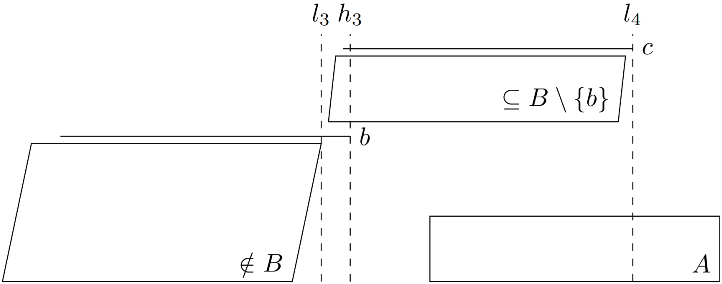
We perform the following steps:

1. We start with $l_4 := l_3 + 1$ and $h_4 := h_3 + 1$
2. We present the new interval $x_i = \frac{l_4 + h_4}{2}$
3. Let j be the chain x_i was assigned to by the algorithm.
   We know that $-1 < x_i < x_B + 1 < 0$, so $j \notin A$ and $j \neq b$
4. Update $l_4 = x_i$
5. If $j \in B$, the we go to round $i + 1$
6. If $j \notin B$, then we move to the next stage

The new intervals form an antichain of size at most $k + 1$. They all had to be assigned to different chains.
None of the those chains are in $A \cup \{b\}$.
At most k of them are in $B \setminus \{b\}$
The stage must finish with an interval x_C assigned to a new chain c.

Stage 4

The new intervals form an antichain of size at most $k+1$. They all had to be assigned to different chains.
None of the those chains are in $A \cup \{b\}$.
At most k of them are in $B \setminus \{b\}$
The stage must finish with an interval x_C assigned to a new chain c.



Stage 5

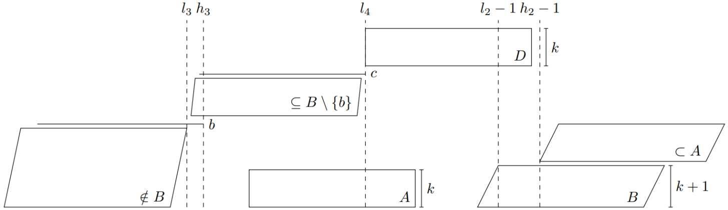
In the last stage we introduce k intervals $x_i = x_C + 1$.

Stage 5

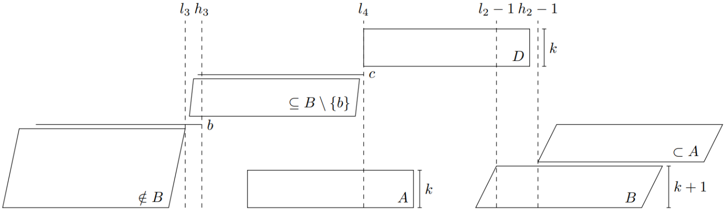
In the last stage we introduce k intervals $x_i = x_C + 1$.
They form an antichain of size k, so the must be assigned to different chain.
None of those chains are in $A \cup B \cup \{c\}$

Stage 5

In the last stage we introduce k intervals $x_i = x_C + 1$.
They form an antichain of size k, so the must be assigned to different chain.
None of those chains are in $A \cup B \cup \{c\}$
We want to show that the width ω has not been exceeded.

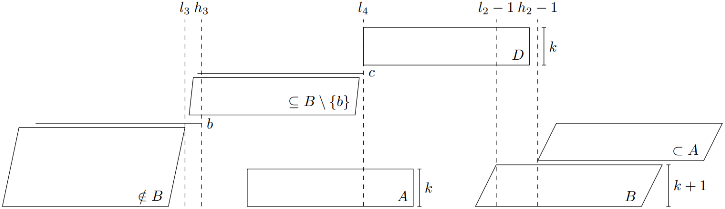


Stage 5



Among intervals introduced in stages 3 and 4, the only interval incomparable to x_i is x_C.

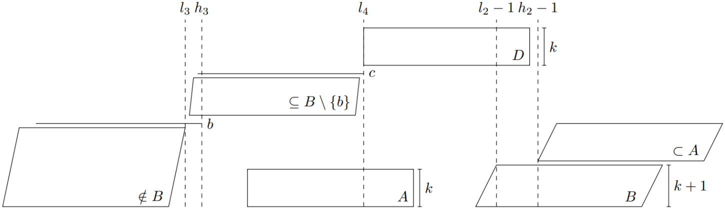
Stage 5



Among intervals introduced in stages 3 and 4, the only interval incomparable to x_i is x_C.
Additionally, we have:

$$l_2 - 3 < x_B < h_2 - 3$$

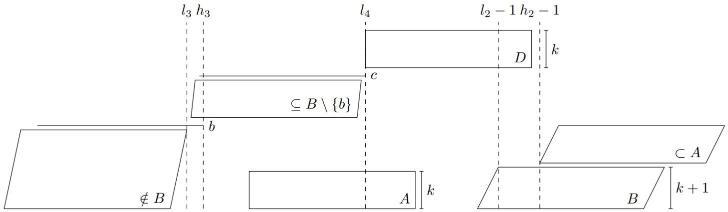
Stage 5



Among intervals introduced in stages 3 and 4, the only interval incomparable to x_i is x_C.
Additionally, we have:

$$l_2 - 3 < x_B < h_2 - 3$$
$$l_2 - 2 < x_C < x_B + 1$$

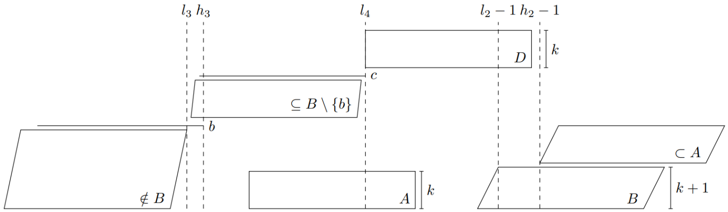
Stage 5



Among intervals introduced in stages 3 and 4, the only interval incomparable to x_i is x_C.
Additionally, we have:

$$l_2 - 3 < x_B < h_2 - 3$$
$$l_2 - 2 < x_C < x_B + 1$$
$$x_i = x_C + 1$$

Stage 5

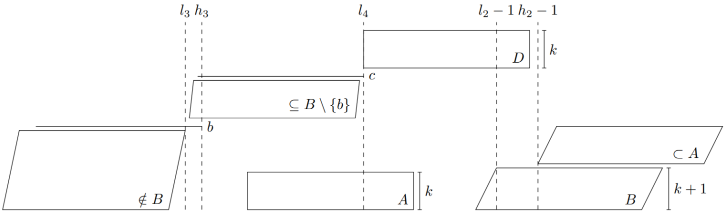


Among intervals introduced in stages 3 and 4, the only interval incomparable to x_i is x_C.
Additionally, we have:

$$l_2 - 3 < x_B < h_2 - 3$$
$$l_2 - 2 < x_C < x_B + 1$$
$$x_i = x_C + 1$$

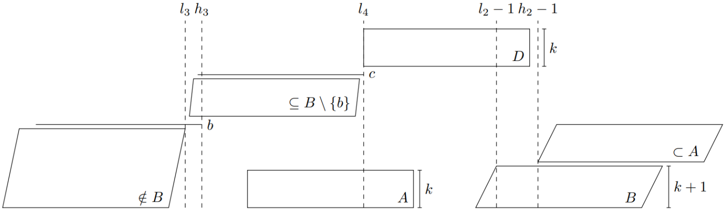
From which we can deduce $l_2 - 1 < x_i < h_2 - 1$. Meaning the intervals from stage 2 incomparable to x_i are exactly the $k + 1$ intervals which were assigned to chains from B.

Stage 5



Let D denote the set of new chains used by the algorithm in stage 5.

Stage 5



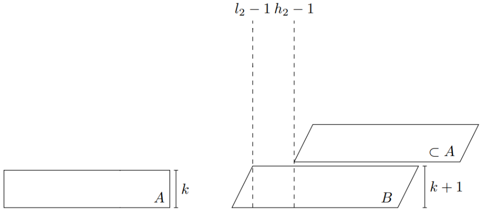
Let D denote the set of new chains used by the algorithm in stage 5. The total number of chains forced by our strategy is thus:

$$|A| + |B| + |\{c\}| + |D| = k + (k+1) + 1 + k = 3k + 2$$

Recap

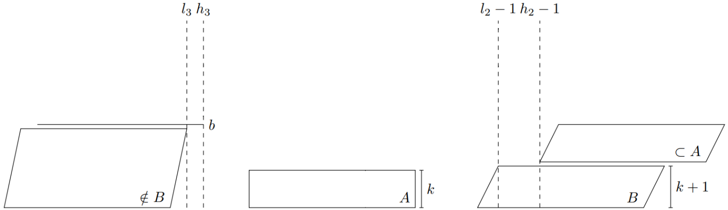
1. We force k chains A with intervals ending at 0.

Recap



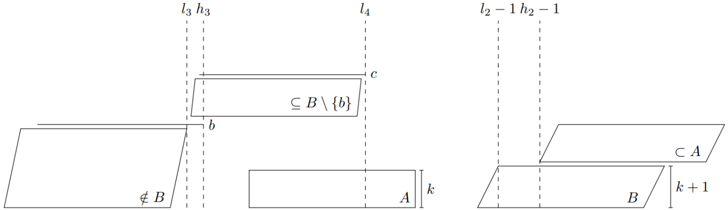
1. We force k chains A with intervals ending at 0.
2. We add new intervals in the $(1,2)$ window until we force $k+1$ new chains - the set B.

Recap



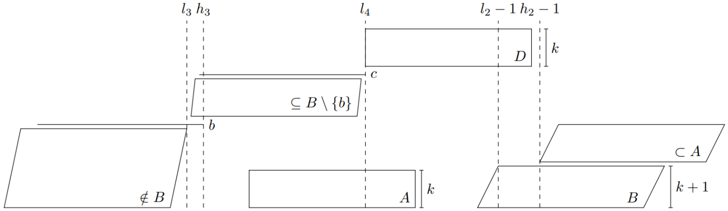
1. We force k chains A with intervals ending at 0.
2. We add new intervals in the $(1, 2)$ window until we force $k + 1$ new chains - the set B.
3. We move the window to the left by 3 and add intervals until one of them is assigned to a chain in B.

Recap



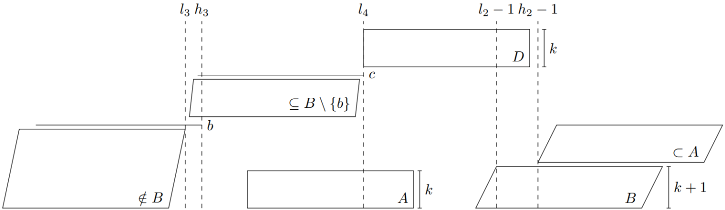
1. We force k chains A with intervals ending at 0.
2. We add new intervals in the $(1, 2)$ window until we force $k + 1$ new chains - the set B.
3. We move the window to the left by 3 and add intervals until one of them is assigned to a chain in B.
4. We move the window by 1 and add intervals until we get a new chain c.

Recap



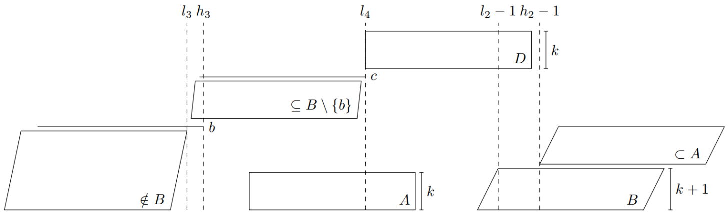
1. We force k chains A with intervals ending at 0.
2. We add new intervals in the $(1, 2)$ window until we force $k + 1$ new chains - the set B.
3. We move the window to the left by 3 and add intervals until one of them is assigned to a chain in B.
4. We move the window by 1 and add intervals until we get a new chain c.
5. We add k new intervals 1 to right of x_C. Each of them forces a new chain, we denote them D.

Conclusion



In the end we forced $3k + 2$ chain in $A \cup B \cup \{c\} \cup D$ while keeping the width of the poset to at most $2k + 1$.

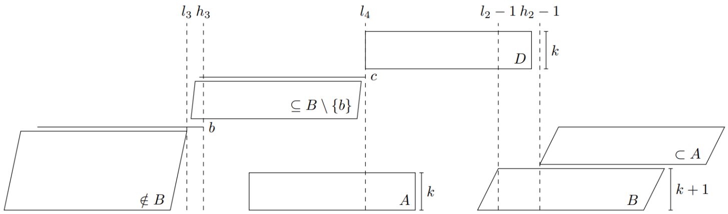
Conclusion



In the end we forced $3k + 2$ chain in $A \cup B \cup \{c\} \cup D$ while keeping the width of the poset to at most $2k + 1$.
That means we have a new best lower bound for the on-line width of semi-orders with representation:

$$\lceil \frac{3}{2}\omega \rceil \leq olws_R(\omega)$$

Conclusion



In the end we forced $3k+2$ chain in $A \cup B \cup \{c\} \cup D$ while keeping the width of the poset to at most $2k+1$.
That means we have a new best lower bound for the on-line width of semi-orders with representation:

$$\lceil \frac{3}{2}\omega \rceil \leq olws_R(\omega) \leq 2\omega - 1$$

By including the upper bound we can give the exact value for $\omega = 3$:

$$olws_R(3) = 5$$

References

Contents and illustrations taken from

[1] C. Biró and I. R. Curbelo, *Improved lower bound on the on-line chain partitioning of semi-orders with representation*, 2021. DOI: 10.48550/ARXIV.2111.04790. [Online]. Available: https://arxiv.org/abs/2111.04790.