

On the on-line coloring of unit interval graphs with proper interval representation

Israel R. Curbelo

Hannah R. Malko

Department of Mathematical Sciences, Kean University, Union, NJ, USA

revisions 21st Aug. 2024, 10th Feb. 2025; accepted 13th Feb. 2025.

We define the problem as a two-player game between Algorithm and Builder. The game is played in rounds. Each round, Builder presents an interval that is neither contained in nor contains any previously presented interval. Algorithm immediately and irrevocably assigns the interval a color that has not been assigned to any interval intersecting it. The set of intervals form an interval representation for a unit interval graph and the colors form a proper coloring of that graph. For every positive integer ω , we define the value $R(\omega)$ as the maximum number of colors for which Builder has a strategy that forces Algorithm to use $R(\omega)$ colors with the restriction that the unit interval graph constructed cannot contain a clique of size $\omega + 1$. In 1981, Chrobak and Ślusarek showed that $R(\omega) \leq 2\omega - 1$. In 2005, Epstein and Levy showed that $R(\omega) \geq \lfloor 3\omega/2 \rfloor$. This problem remained unsolved for $\omega \geq 3$. In 2023, Biró and Curbelo showed that $R(3) = 5$. In this paper, we show that $R(4) = 7$.

Keywords: online algorithms, online coloring, interval graphs, graph coloring

1 Introduction

An on-line graph coloring algorithm A receives a graph G in the order of its vertices v_1, \dots, v_n and constructs an on-line coloring. This means that the color assigned to a vertex v_i depends solely on the subgraph induced by the vertices $\{v_1, \dots, v_i\}$ and on the colors assigned to them. The simplest on-line coloring algorithm is the greedy algorithm, First-Fit, which traverses the list of vertices given in the order they are presented, and assigns each vertex the minimal color not assigned to its neighbors that appear before it in the list of vertices.

The performance of an algorithm A on a graph G is measured with respect to the chromatic number $\chi(G)$, which is equivalent to the number of colors needed by an optimal off-line algorithm. Let $\chi_A(G)$ be the maximum number of colors needed by algorithm A over all orderings of the vertices of G . An on-line coloring algorithm A is *strictly ρ -competitive* if for every graph G , $\chi_A(G) \leq \rho \cdot \chi(G)$. The *strict competitive ratio* of algorithm A is $\inf\{\rho \mid A \text{ is strictly } \rho\text{-competitive}\}$. In general, First-Fit does not have a bounded competitive ratio, and there is no on-line algorithm with a constant competitive ratio. The best known algorithm by Halldórsson (1997); Halldórsson and Szegedy (1994) uses $O(\chi \cdot n / \log n)$ colors for χ -colorable graphs of size n , and no on-line coloring algorithm can be $o(n / \log^2 n)$ -competitive. For this reason, it is common to restrict the class of graphs. An important class of intersection graphs that has been extensively studied in this context, with applications in scheduling, is the class of interval graphs.

A graph $G = (V, E)$ is an *interval graph* if there is a function f which assigns to each vertex $v \in V$ a closed interval $f(v)$ on the real line so that for all $u, v \in G$, u and v are adjacent if and only if $f(u) \cap f(v) \neq \emptyset$. We refer to f an *interval representation* of G . When no interval is properly contained in any other interval, we refer to G as a *proper interval graph* and f as a *proper interval representation* of G . When every interval has unit length, we refer to G as a *unit interval graph* and f as a *unit interval representation* of G . Since every unit interval representation is a proper interval representation, any unit interval graph is a proper interval graph. Moreover, it can be shown that an interval graph G is a unit interval graph if and only if G is a proper interval graph.

The on-line coloring problem for the class of interval graphs was solved by Kierstead and Trotter (1981) when they constructed an on-line algorithm with competitive ratio 3, and proved that no on-line algorithm could have a better competitive ratio. Chrobak and Ślusarek (1988) proved the same result, but instead of the vertices of the graph being presented as points, the graph was presented with its interval representation. Naturally, this leads us to question if knowledge of the interval representation makes a difference in the problem. Chrobak and Ślusarek also considered the problem restricted to unit interval graphs with known representation. They showed that First-Fit has a competitive ratio of 2. Since First-Fit does not use information from the representation, this holds true for both cases. However, the question of whether a better online algorithm exists remained unanswered. History would have us believe that there should be a more efficient algorithm. The most closely related example would be the performance of First-Fit on the class of interval graphs. Woodall (1974) was the first to consider this problem. After extensive research on this problem as seen in Kierstead (1988); Kierstead and Qin (1995); Kierstead and Trotter (1981); Chrobak and Ślusarek (1988); Pemmaraju et al. (2004); Narayanaswamy and Subhash Babu (2008); Kierstead et al. (2016), it was found that the competitive ratio of the First-Fit algorithm for interval graphs is between 5 and 8. This is worse than the optimal algorithm developed by Kierstead and Trotter (1981), which has a competitive ratio of 3.

Epstein and Levy (2005) presented a strategy for constructing a unit/proper interval graph with representation that would force any on-line algorithm to use $3k$ colors on a graph with clique size k . Hence, they showed that there is no on-line algorithm with competitive ratio less than $\frac{3}{2}$. In 2008, Micek studied this problem in terms of on-line chain partitioning and showed that there is no algorithm better than First-Fit if the graph is presented without representation. (We refer the reader to the survey paper by Bosek et al. (2012) for a comprehensive overview of these problems in terms of on-line chain partitioning and for a proof of this result.) Specifically, there is no on-line algorithm with competitive ratio better than 2. Micek also claimed that the strategy by Epstein and Levy (2005) could be improved if a proper interval representation was used. Biró and Curbelo (2023) showed that the algorithm could also be improved for unit intervals. This provided a tight bound of 5 when the chromatic number is restricted to at most 3 and showed that there is no on-line algorithm with competitive ratio less than $\frac{5}{3}$ with unit or proper representation. It remains unknown whether knowledge of the representation or if the choice of representation makes a difference to the result of the problem.

In this paper, we present a strategy which forces any on-line algorithm to use 7 colors on a unit interval graph with chromatic number at most 4 and known proper interval representation and prove the following.

Theorem 1.1 *There is no on-line algorithm with strict competitive ratio less than $\frac{7}{4}$ for the on-line coloring problem restricted to unit interval graphs with known proper interval representation.*

2 Preliminaries

We define the problem as a two-player game between Builder and Algorithm. The game is played in rounds. Each round, Builder presents an interval so that any real number contained in that interval is contained in at most three other intervals. Algorithm, immediately and irrevocably, assigns the interval a color from the set $\mathcal{G} = \{a, b, c, d, e, f, g\}$ with the restriction that any two intersecting intervals cannot be assigned the same color. Anytime that Algorithm uses a new color, we may assume that it is the first unused color alphabetically in \mathcal{G} . In this paper, we play the role of Builder, attempting to force Algorithm to use the color g . We restrict the domain of the game so that all intervals presented by Builder must be properly contained in the interval $[0, 1]$. At the end of round i , Builder may further restrict the domain by choosing two real numbers l_i and r_i so that $l_{i-1} \leq l_i < r_i \leq r_{i-1}$ and any interval presented after round i must be properly contained in $[l_i, r_i]$. We refer to l_i and r_i as *walls*. Note that this restriction makes no difference to the outcome of the game.

2.1 State representation

Consider a sequence of intervals x_1, \dots, x_n presented by Builder and a sequence of corresponding colors y_1, \dots, y_n assigned by Algorithm. The set of interval-color pairs

$$S = \{(x_i, y_i) \mid i \in \{1, \dots, n\}\}$$

defines a *state*.

Let S be a state as defined above. We define its *state representation matrix* by

$$f(S) = \begin{bmatrix} s_1 & s_2 & \dots & s_{2n} \\ z_1 & z_2 & \dots & z_{2n} \end{bmatrix}$$

where the sequences s_1, \dots, s_{2n} and z_1, \dots, z_{2n} are obtained as follows. First, let r_1, \dots, r_{2n} be obtained by taking all $2n$ endpoints of x_1, \dots, x_n and ordering them in ascending order. Next, for $i \in \{1, \dots, 2n\}$, $s_i = 0$ if r_i was a left endpoint and $s_i = 1$ if r_i was a right endpoint. Finally, z_i denotes the color in \mathcal{G} that was assigned to the interval which r_i belonged to.

We consider two states equivalent if they have the same state representation matrix. Furthermore, we consider a state T equivalent to S if and only if there exists a permutation $\sigma : \mathcal{G} \rightarrow \mathcal{G}$ so that

$$f(T) = \begin{bmatrix} s_1 & s_2 & \dots & s_{2n} \\ \sigma z_1 & \sigma z_2 & \dots & \sigma z_{2n} \end{bmatrix}.$$

The following statement allows us to generalize and represent states visually throughout the paper.

Proposition 2.1 *Let S_1 and S_2 be states. If S_1 is equivalent to S_2 and there is a winning strategy for Builder starting at state S_1 , then there is a winning strategy for Builder starting at state S_2 .*

Given a state S , let S^* denote the *dual* of S defined as a state constructed by reversing the order of the columns in the state representation matrix of S . The following statement allows us switch between a state and its dual in our strategy.

Proposition 2.2 *Let S be a state. If there is a winning strategy for Builder starting at state S , then there is a winning strategy for Builder starting at state S^* .*

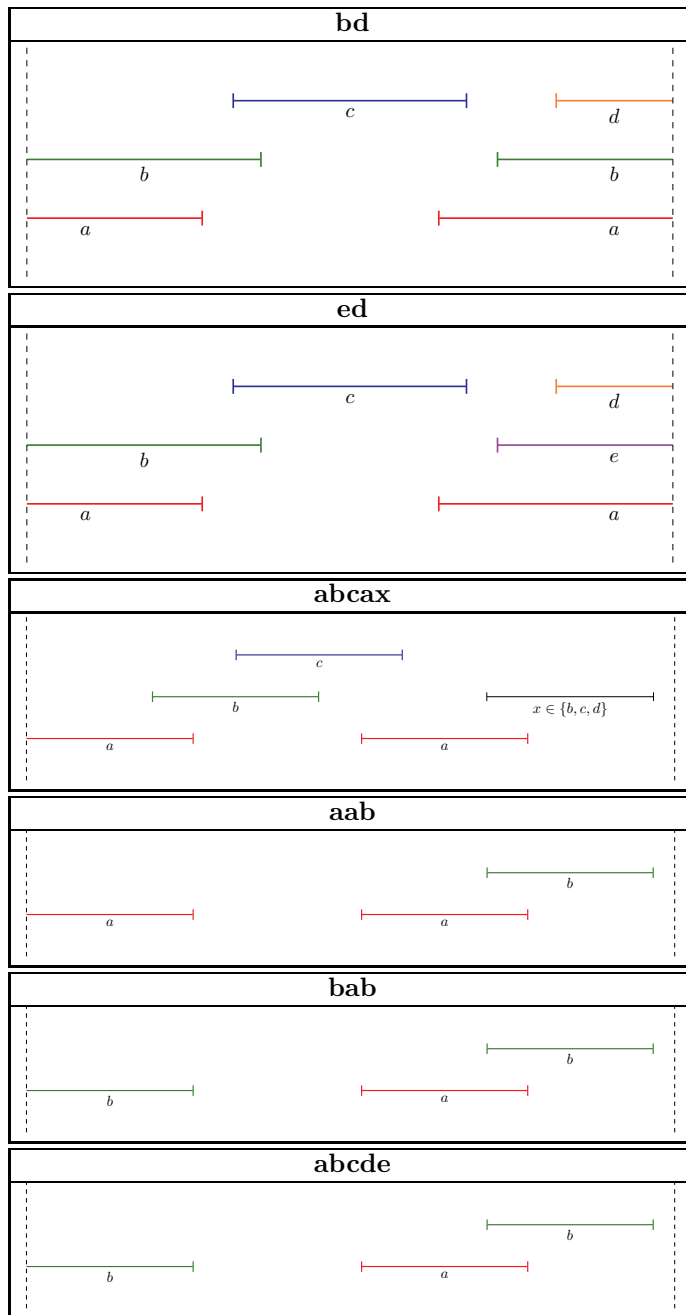


Fig. 1: States

To further generalize states to includes walls, we simply ignore endpoints outside of the walls. For example, we say that a state S is a **bd** state (shown in Figure 1) if there exists a permutation $\sigma : \mathcal{G} \rightarrow \mathcal{G}$ such that

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ \sigma a & \sigma c & \sigma b & \sigma a & \sigma c & \sigma b & \sigma d \end{bmatrix}$$

is a submatrix of contiguous columns of $f(S)$. Additionally, we say that a state S is a **bd*** state if there exists a permutation $\sigma : \mathcal{G} \rightarrow \mathcal{G}$ such that

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ \sigma d & \sigma b & \sigma c & \sigma a & \sigma b & \sigma c & \sigma a \end{bmatrix}$$

is a submatrix of contiguous columns of $f(S^*)$, that is, if S^* is a **bd** state. For convenience, we define a **gg** state as any state that contains all seven colors in \mathcal{G} .

2.2 Separation strategy

We generalize the separation strategy used by Epstein and Levy (2005) and provide a proof of the following lemma for completeness.

Lemma 2.3 *Let $[\alpha_1, \beta_1]$ and $[\alpha_2, \beta_2]$ be intervals with $\beta_1 < \alpha_2$, let Y be a set of colors, and let k be a positive integer. Then for every on-line algorithm there is a strategy for Builder to present k intervals so that they will form a set $\{[l_1, r_1], \dots, [l_k, r_k]\}$ with $\alpha_1 < l_1 < \dots < l_k < \beta_1 < \alpha_2 < r_1 < \dots < r_k < \beta_2$, and there is an integer $j \in \{0, \dots, k\}$ such that for every $i \in \{1, \dots, k\}$, Algorithm assigns the interval $[l_i, r_i]$ a color in Y if and only if $i \leq j$.*

Proof: Let $[\alpha_1, \beta_1]$ and $[\alpha_2, \beta_2]$ be intervals with $\beta_1 < \alpha_2$, and let Y be a set of colors. We argue by induction on the positive integer k . If $k = 1$, then we present the interval $I = [(\alpha_1 + \beta_1)/2, (\alpha_2 + \beta_2)/2]$. If Algorithm assigns the interval I a color in Y , then $j = 1$. Otherwise, $j = 0$.

Suppose $k > 1$. By the induction hypothesis, there exists a strategy for Builder to present $k - 1$ intervals so that they will form a set $\{[l'_1, r'_1], \dots, [l'_{k-1}, r'_{k-1}]\}$ with $\alpha_1 < l'_1 < \dots < l'_{k-1} < \beta_1 < \alpha_2 < r'_1 < \dots < r'_{k-1} < \beta_2$ and a integer $j' \in \{0, \dots, k - 1\}$ such that for every $i \in \{1, \dots, k - 1\}$, Algorithm assigns the interval $[l'_i, r'_i]$ a color from Y if and only if $i \leq j'$. If $j' = 0$, we introduce the interval $I = [(\alpha_1 + l'_1)/2, (\alpha_2 + r'_1)/2]$ and define $[l_i, r_i] = [l'_{i-1}, r'_{i-1}]$ for $i \in \{2, \dots, k\}$ and $[l_1, r_1] = I$. If $j' = k - 1$, we introduce the interval $I = [(l'_{k-1} + \beta_1)/2, (r'_{k-1} + \beta_2)/2]$ and define $[l_i, r_i] = [l'_i, r'_i]$ for $i \in \{1, \dots, k - 1\}$ and $[l_k, r_k] = I$. Otherwise, we introduce the interval $I = [(l'_{j'} + l'_{j'+1})/2, (r'_{j'} + r'_{j'+1})/2]$ and define $[l_i, r_i] = [l'_i, r'_i]$ for $i \in \{1, \dots, j\}$, $[l_i, r_i] = [l'_{i-1}, r'_{i-1}]$ for $i \in \{j + 2, \dots, k\}$ and $[l_{j+1}, r_{j+1}] = I$. If Algorithm assigns the new interval I a color in Y , then $j = j' + 1$. Otherwise, $j = j'$. \square

Lemma 2.3 implies that if S is a state with state representation matrix

$$f(S) = \begin{bmatrix} s_1 & s_2 & \dots & s_{2n} \\ z_1 & z_2 & \dots & z_{2n} \end{bmatrix},$$

then for any positive integers p and q with $1 \leq p < q \leq 2n$, positive integer k and set of colors $Y \subseteq \mathcal{G}$, we can introduce k intervals so that the result is a state S' with state representation matrix

$$f(S') = \begin{bmatrix} s_1 \dots s_{p-1} & 0 \dots 0 & s_p \dots s_q & 1 \dots 1 & s_{q+1} \dots s_{2n} \\ z_1 \dots z_{p-1} & z'_1 \dots z'_k & z_p \dots z_q & z'_1 \dots z'_k & z_{q+1} \dots z_{2n} \end{bmatrix}$$

and there is an integer $j \in \{0, \dots, k\}$ such that for every $i \in \{1, \dots, k\}$, $z'_i \in Y$ if and only if $i \leq j$. Note that when $p = 1$, the first column of $f(S')$ is $(0, z'_1)$, and when $q = 2n$, the last column of $f(S')$ is $(1, z'_k)$.

We use this separation strategy throughout the proof of Theorem 1.1 in the form of “we present k intervals as shown in Figure t while separating any interval assigned a color in Y to the left” or equivalently “we present k intervals as shown in Figure t while separating any interval assigned a color in $\mathcal{G} - Y$ to the right” with t denoting the Figure that defines $[\alpha_1, \beta_1]$ and $[\alpha_2, \beta_2]$ by the nearest endpoints to the left and to the right of the left and right endpoints of the first new interval, respectively. When $Y = \{y\}$ for some $y \in \mathcal{G}$, we simply state “assigned the color y ” instead of “assigned a color in Y ”.

2.3 Outline of proof

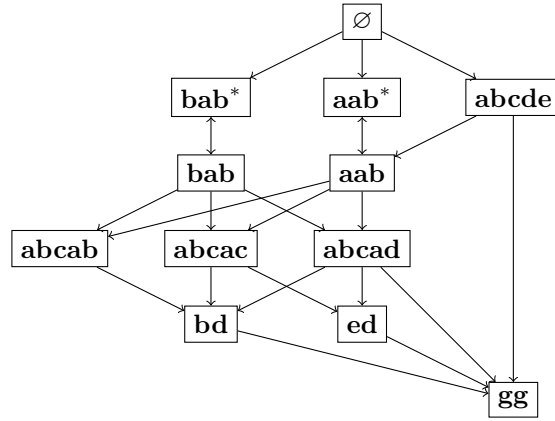


Fig. 2: Outline of the proof of Theorem 1.1.

In Section 3, we prove Theorem 1.1 by showing that there is a strategy for Builder which forces a **gg** state starting from each of the states in Figure 1 and then showing that we can always force at least one of those states or the dual of one of those states. Note that the third state **abcax** in Figure 1 contains an interval labeled with an arbitrary color $x \in \{b, c, d\}$. This is to represent three states **abcab**, **abcac** and **abcad** where the interval is colored b , c and d , respectively. That is, a state is an **abcax** state if and only if it is an **abcab** state, an **abcac** state or an **abcad** state.

The structure of the proof is as follows. In Section 3.1 and Section 3.2, we show that we can force a **gg** state starting from any **bd** state or from any **ed** state, respectively. In Section 3.3, Section 3.4 and Section 3.5, we show that that we can force a **gg** state starting from any **abcax** state. In particular, in Section 3.3, we show that we can force a **bd** state starting from any **abcab** state, and in Section 3.4, we show that we can force a **bd** or an **ed** state starting from any **abcac** state. In Section 3.5, we show that we can force a **gg** state starting from any **abcad**. In Section 3.6, we show that we can force an **abcax** state starting from either any **aab** state or any **bab** state. In Section 3.7, we show that we can force a **gg** state starting from any **abcde** state. Finally, in Section 3.8, we show that we can always force either an **aab*** state, a **bab*** state or an **abcde** state. The complete outline is summarized in Figure 2.

3 Proof of Main Theorem

3.1 (bd → gg)

Assume we start in a **bd** state. Figure 3 shows how to force a **gg** state in two phases. In the first phase, we present two intervals as shown in Figure 3 while separating any interval assigned the color d to the left. Let x be the color assigned to the interval on the left and y be the color assigned to the interval on the right. In the second phase, there are two cases. That is, either Algorithm used the color d in the first phase or Algorithm did not use the color d in the first phase. If Algorithm used the color d in the first phase, then $x = d$ and $y = e$. The bottom left frame of Figure 3 shows how to force a **gg** state in two moves. If neither interval was assigned the color d , then both intervals were assigned new colors. Without loss of generality, let us assume that $x = f$ and $y = e$. The bottom right frame of Figure 3 shows how to force a **gg** state in one move.

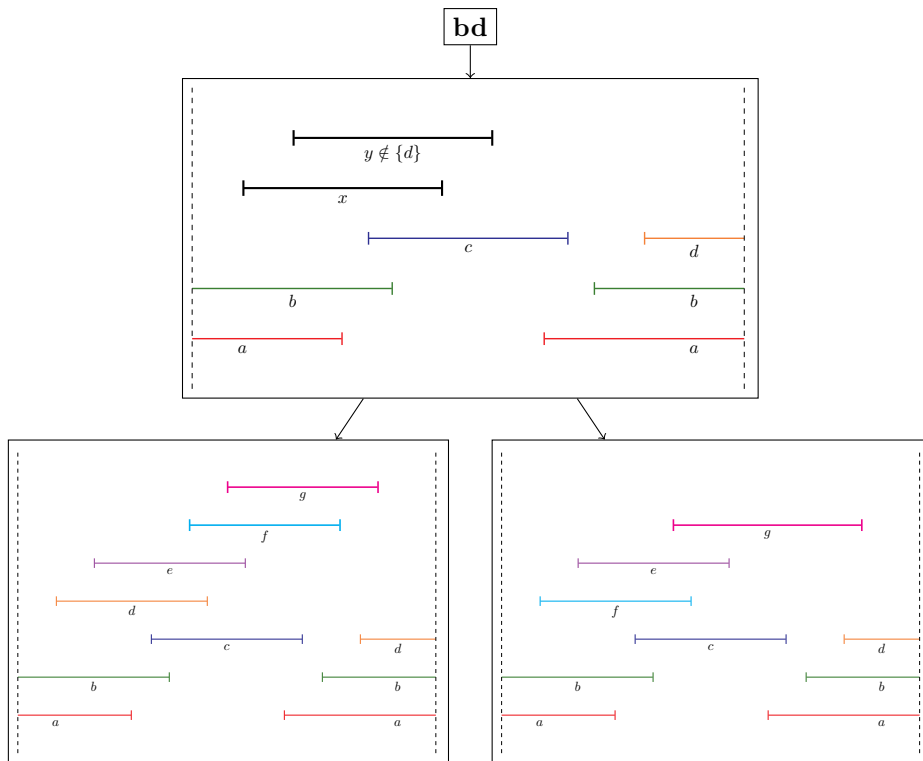


Fig. 3: (bd → gg)

3.2 (ed → gg)

Assume we start in an **ed** state. Figure 4 shows how to force a **gg** state in two phases. In the first phase, we present two intervals as shown in Figure 4 while separating any interval assigned the color e to the

left. Let y denote the color assigned to the right interval. Since $y \neq e$, it must be the case that $y \in \{d, f\}$, and hence, there are two cases in the second phase. If $y = d$, then the bottom left frame of Figure 3 shows how to force a **gg** state in two moves. If $y = f$, the bottom right frame of Figure 4 shows how to force a **gg** state in one move.

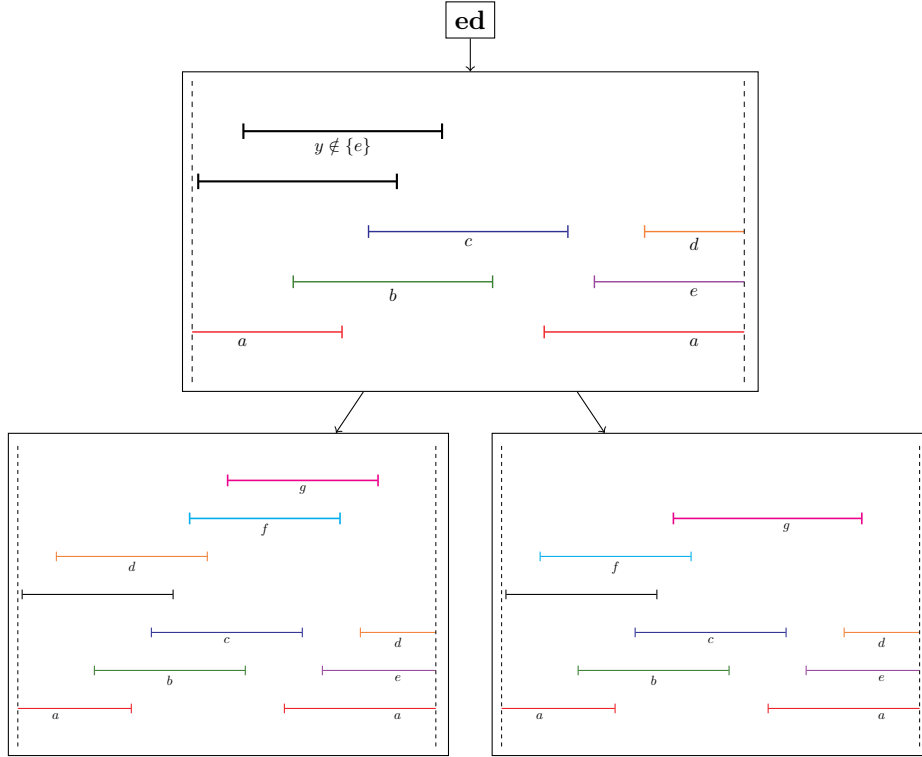


Fig. 4: **ed** \rightarrow **gg**

3.3 (**abcab** \rightarrow **bd**)

Assume we start in an **abcab** state. Figure 5 shows how to force a **bd** state in at most two moves. We present two intervals and update the right wall as shown in Figure 5 while separating any interval that was assigned the color c to the right. Let x denote the color that Algorithm assigned to the left interval. Since neither interval could have been assigned the colors a or b , the left interval must have been colored with a new color. Without loss of generality, we may assume that $x = d$, resulting in a **bd** state.

3.4 (**abcac** \rightarrow **bd** \vee **ed**)

Assume we start in an **abcac** state. Figure 6 shows how we can force either a **bd** state or an **ed** state in at most two moves. We present two intervals and update the right wall as shown in Figure 6 while separating any interval assigned the color b to the left. Let x denote the color assigned to the left interval

and y denote the color assigned to the right interval. If Algorithm assigned the color b to either of the two intervals, then $x = b$ and $y = d$ which results in a bd state. If Algorithm did not assign the color b to either interval, then both intervals were assigned a new color. Without loss of generality, we may assume $x = e$ and $y = d$ resulting in an ed state.

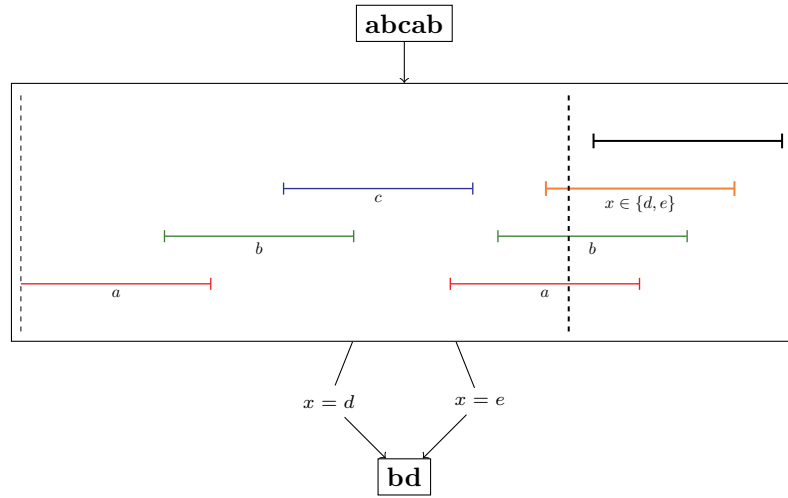


Fig. 5: $abcab \rightarrow bd$

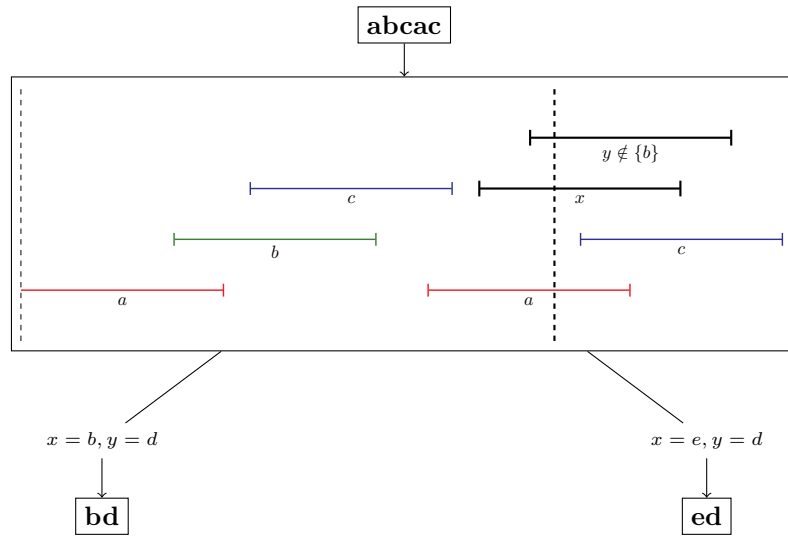


Fig. 6: $abcac \rightarrow bd \vee ed$

3.5 $\text{abcd} \rightarrow \text{gg}$

Assume we start in an abcd state. We present one interval as in Figure 7. Let x denote the color assigned to the interval by Algorithm. Then, $x \in \{b, c, e\}$. If $x = e$, then the result is an ed state. If $x = c$, then notice that if we introduce a new interval between the interval colored d and the interval colored $x = e$, then updating the right wall to exclude only the interval colored $x = e$ results in either a bd state or an ed state. Hence, we may assume that $x = b$.

We present two intervals as shown in Figure 7 while separating any interval assigned the color a to the right. Let y denote the color assigned to the left interval by Algorithm. Then, $y \in \{c, e, f\}$. If $y = c$, then notice that if we introduce a new interval $[l, r]$ with l immediately to the left of the left endpoint of the interval colored d and r immediately to the right of the left endpoint of the interval colored $y = c$, then updating the right wall to be immediately to the right of the left endpoint of the interval colored d results in an ed state. Hence, we may assume that the interval was colored a new color. Without loss of generality, let us assume that $y = e$.

We present two intervals as shown in Figure 7 while separating any interval assigned the color d to the left. Let z denote the color that was assigned to the right interval by Algorithm. Then $z \in \{e, f\}$. If $z = e$, then Figure 8 shows how to force colors f and g in two moves. Therefore, we may assume that $z = f$. We present an interval as shown in Figure 8. Let w denote the color assigned to the interval by Algorithm. Then, $w \in \{e, f\}$. Regardless of the choice of color by Algorithm, we can force the color g in one move. Figure 8 shows how to force g in one move for each case. Thus, we have shown that we can force a gg state from any abcd state.

3.6 $(\text{aab} \vee \text{bab}) \rightarrow \text{abcax}$

Assume we start in an aab state. Figure 9 shows how to force an abcax state from an aab state. We present two intervals as shown in Figure 9. The possibilities for the two colors are (b, c) , (c, b) , (c, d) and (d, c) . If the intervals are colored b and c , then the result is an abcab state, If the intervals are colored c and b , then the result is an abcac state, and if the intervals are colored c and d , or d and c , then the result is an abcd state. Hence, we can force an abcax state from any aab state.

Assume we start in a bab state. Figure 10 shows how to force an abcax state from a bab state. We present two intervals as shown in Figure 10. The first interval on the left must be colored c . The possible colors for the second interval are a , c and d . If the interval is colored a , then the result is an abcac state, if the interval is colored c , then the result is an abcab state, and if the interval is colored d , then the result is an abcd state. Hence, we can force an abcax state from any bab state. Since we can force g from any abcax state, we can force a gg state from any aab state and from any bab state.

3.7 $(\text{abcde}) \rightarrow \text{gg}$

Assume we start in an abcde state. Figure 11 shows how to force a gg state in three phases. In the first phase, we present four intervals while separating any interval assigned a color in $\{b, e, f\}$ to the left. The right-most interval must be colored either a , c or d . If the interval is colored a , then the result is an aab state. Hence, we may assume that the interval was colored c or d . In the second phase, we present two intervals while separating any interval assigned a color in $\{e, f\}$ to the right. It is easy to see that in order to avoid using g , the interval on the right must be colored either e or f . The bottom two frames of Figure 11 show how to force a gg state for each of the two cases.

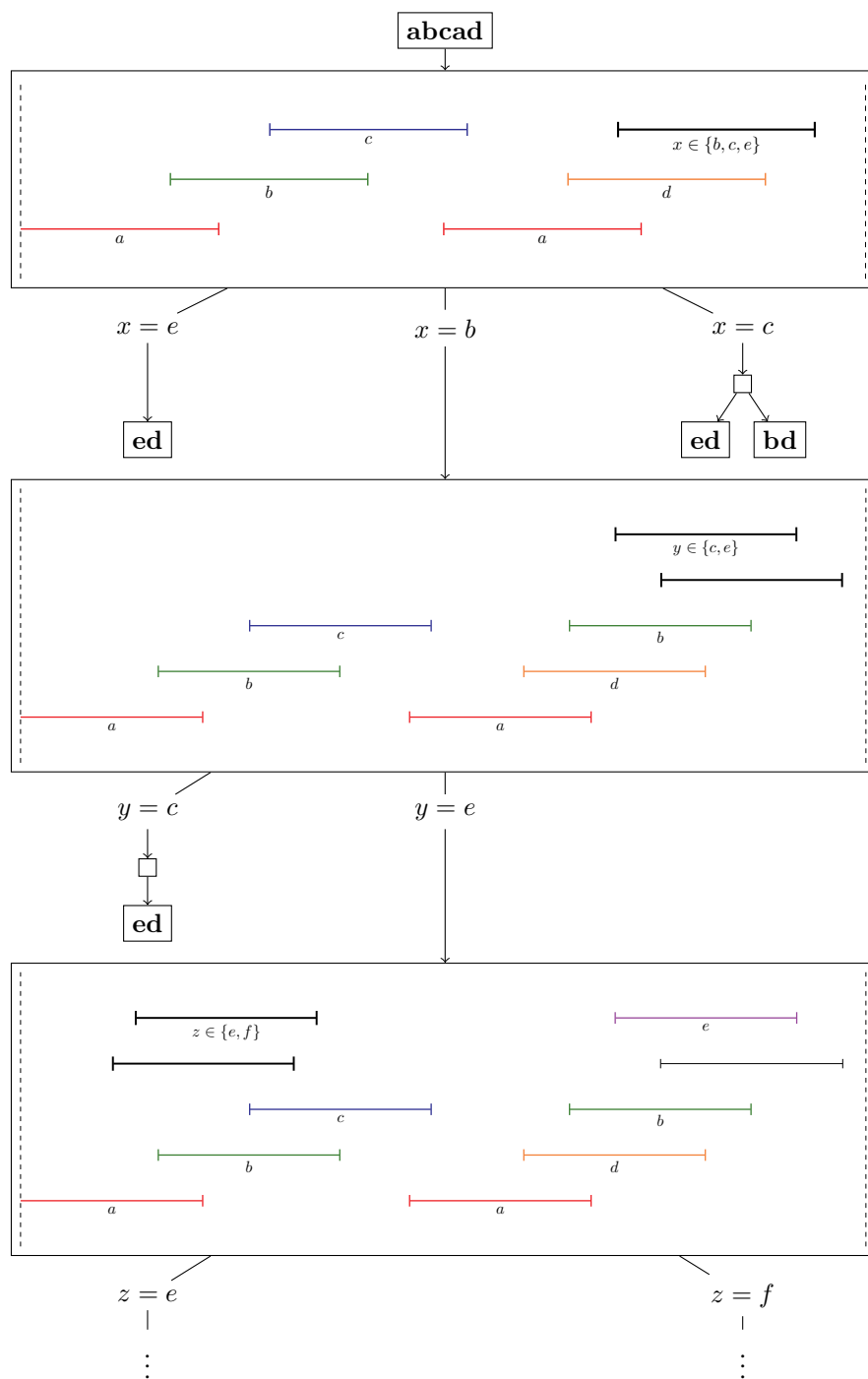


Fig. 7: $abcd \rightarrow gg$

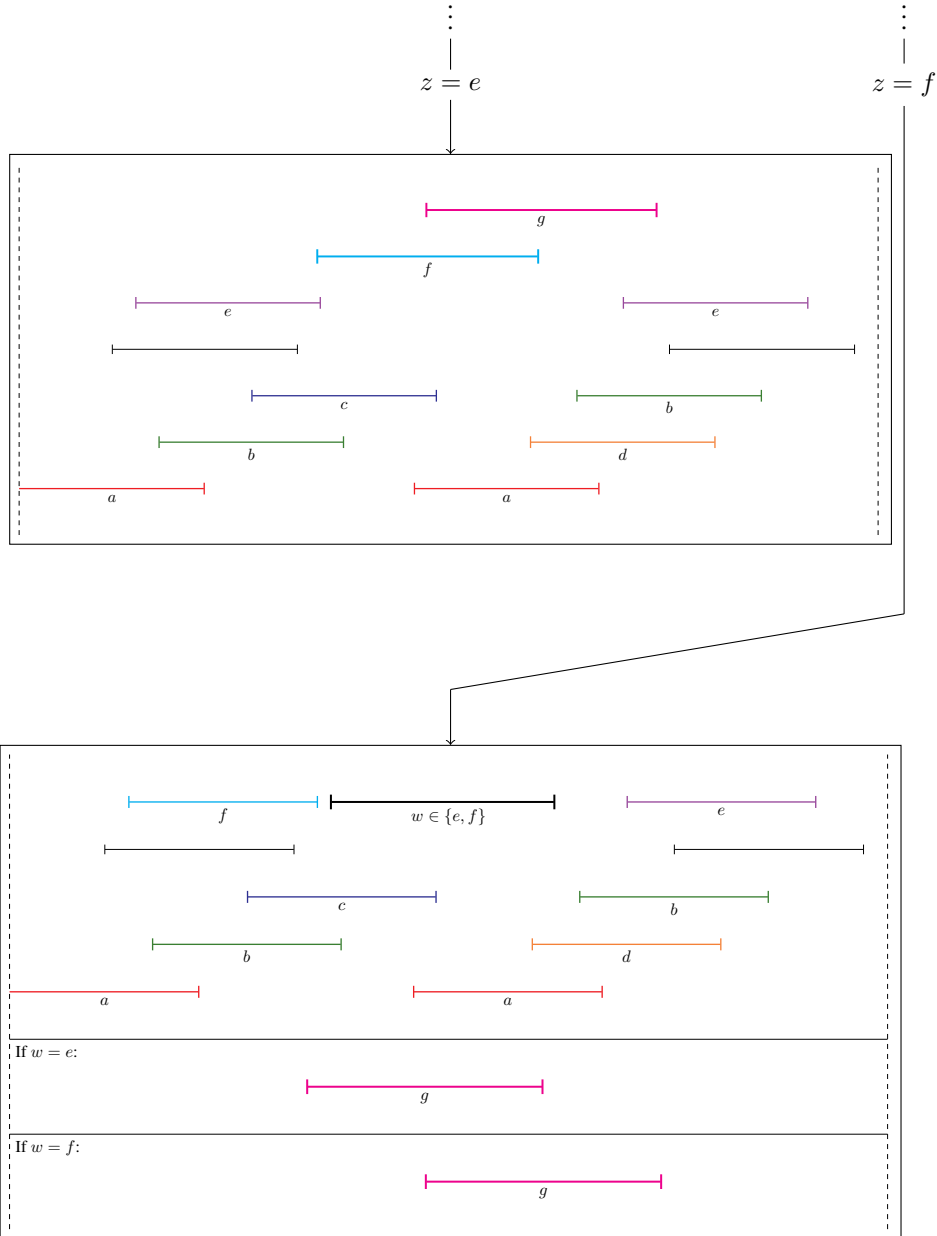


Fig. 8: $abcd \rightarrow gg$ (continued.)

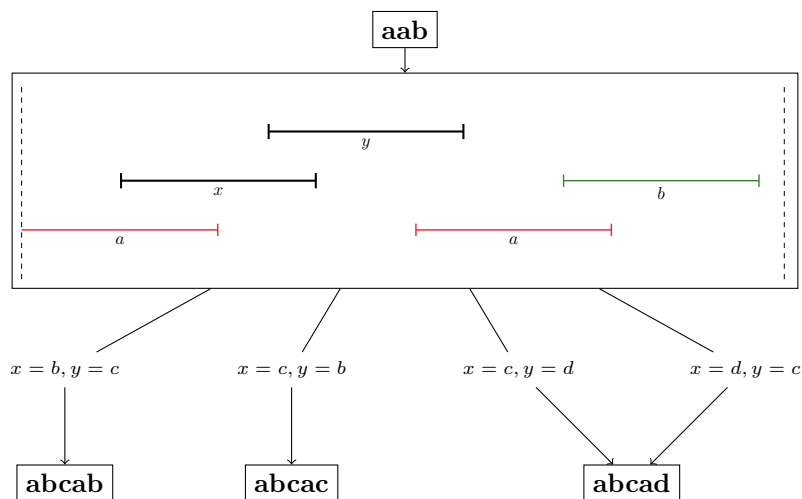


Fig. 9: $aab \rightarrow abcax$

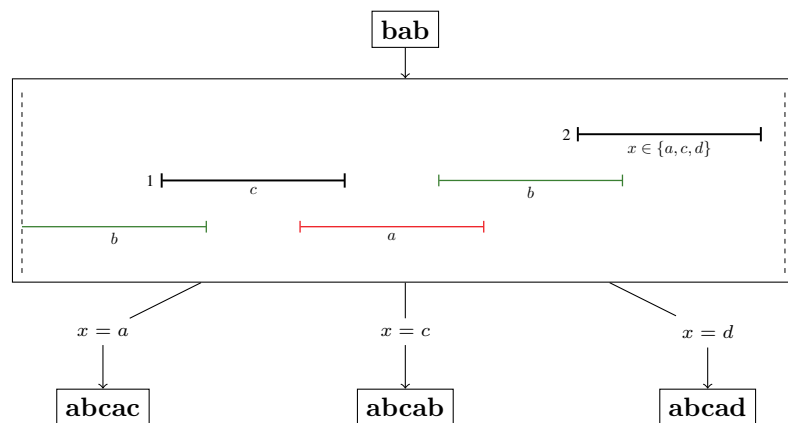


Fig. 10: $bab \rightarrow abcax$

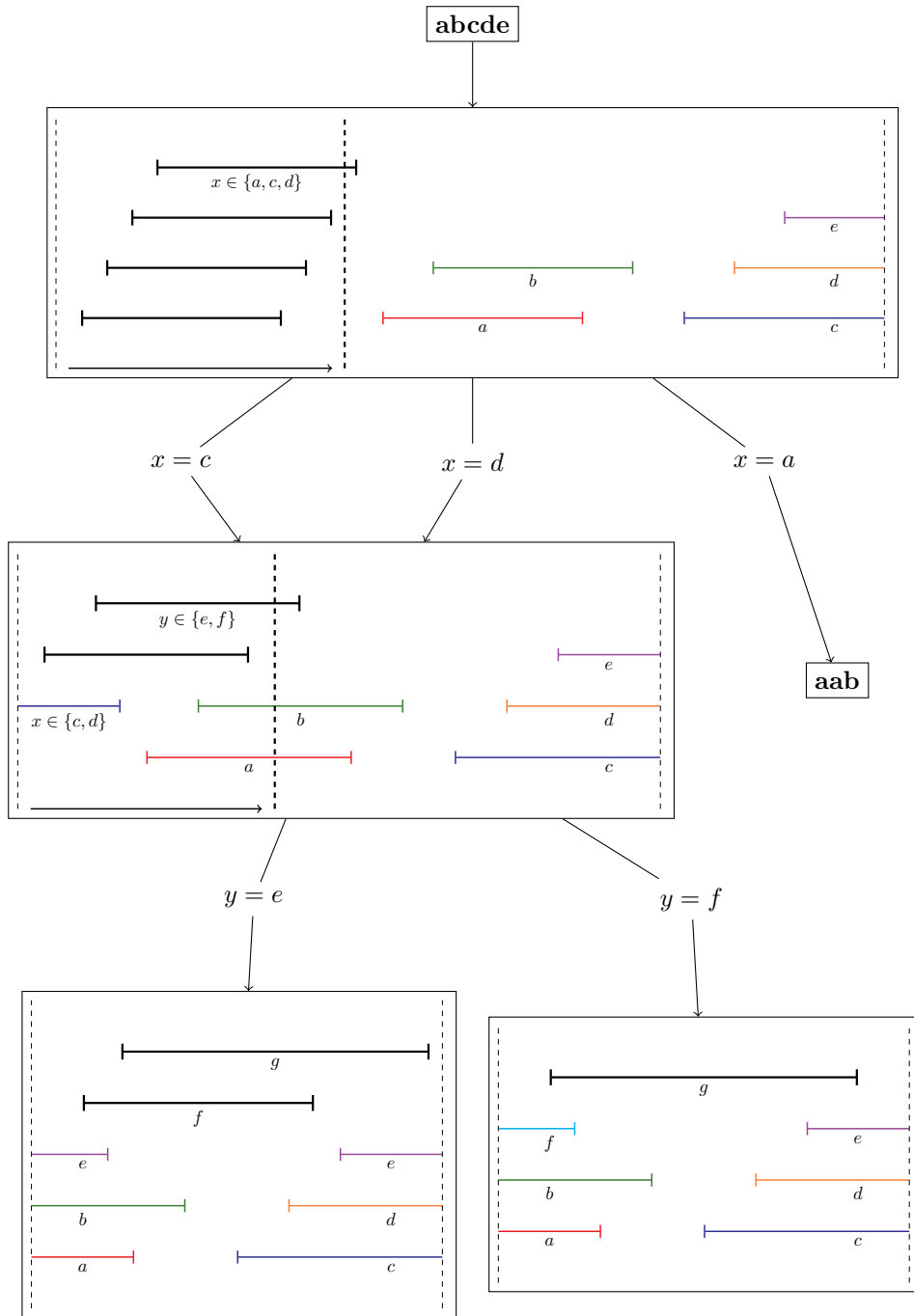


Fig. 11: $abcde \rightarrow aab \vee gg$

3.8 (gg)

Fig. 12: $\mathbf{aab}^* \vee \mathbf{bab}^* \vee \mathbf{abcde}$

Finally, we show that we can always force an \mathbf{aab}^* state, a \mathbf{bab}^* state or an \mathbf{abcde} state. To show this, we simply introduce intervals in the order shown in Figure 12. Intervals 1 and 2 must be assigned colors a and b , respectively. When presented, intervals 3, 4 and 5 cannot be assigned the color a since the result would be an \mathbf{aab}^* state, nor the color b since the result would be a \mathbf{bab}^* state (which can be seen by updating the right wall to be immediately to the right of the left endpoint of the newest interval). Thus, the result after presenting all five intervals is an \mathbf{abcde} state. This concludes the proof.

References

- C. Biró and I. R. Curbelo. Improved lower bound on the on-line chain partitioning of semi-orders with representation. *Discrete Math.*, 346(12):Paper No. 113656, 7, 2023. ISSN 0012-365X,1872-681X. doi: 10.1016/j.disc.2023.113656.
- B. Bosek, S. Felsner, K. Kloch, T. Krawczyk, G. Matecki, and P. Micek. On-line chain partitions of orders: a survey. *Order*, 29(1):49–73, 2012. ISSN 0167-8094. doi: 10.1007/s11083-011-9197-1.
- M. Chrobak and M. Ślusarek. On some packing problem related to dynamic storage allocation. *RAIRO Inform. Théor. Appl.*, 22(4):487–499, 1988. ISSN 0296-1598.
- L. Epstein and M. Levy. Online interval coloring and variants. In *Automata, languages and programming*, volume 3580 of *Lecture Notes in Comput. Sci.*, pages 602–613. Springer, Berlin, 2005. doi: 10.1007/11523468_49.
- M. M. Halldórsson. Parallel and on-line graph coloring. *J. Algorithms*, 23(2):265–280, 1997. ISSN 0196-6774. doi: 10.1006/jagm.1996.0836.
- M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. *Theoret. Comput. Sci.*, 130(1):163–174, 1994. ISSN 0304-3975,1879-2294. doi: 10.1016/0304-3975(94)90157-0.
- H. A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM J. Discrete Math.*, 1(4): 526–530, 1988. ISSN 0895-4801. doi: 10.1137/0401048.
- H. A. Kierstead and J. Qin. Coloring interval graphs with First-Fit. *Discrete Math.*, 144(1-3):47–57, 1995. ISSN 0012-365X. doi: 10.1016/0012-365X(94)00285-Q. *Combinatorics of ordered sets* (Oberwolfach, 1991).

- H. A. Kierstead and W. T. Trotter, Jr. An extremal problem in recursive combinatorics. *Congr. Numer.*, 33:143–153, 1981. ISSN 0384-9864.
- H. A. Kierstead, D. A. Smith, and W. T. Trotter. First-fit coloring on interval graphs has performance ratio at least 5. *European J. Combin.*, 51:236–254, 2016. ISSN 0195-6698. doi: 10.1016/j.ejc.2015.05.015.
- N. S. Narayanaswamy and R. Subhash Babu. A note on first-fit coloring of interval graphs. *Order*, 25(1): 49–53, 2008. ISSN 0167-8094. doi: 10.1007/s11083-008-9076-6.
- S. V. Pemmaraju, R. Raman, and K. Varadarajan. Buffer minimization using max-coloring. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 562–571. ACM, New York, 2004.
- D. R. Woodall. Applications of polymatroids and linear programming to transversals and graphs. In *Combinatorics (Proc. British Combinatorial Conf., Univ. Coll. Wales, Aberystwyth, 1973)*, London Math. Soc. Lecture Note Ser., No. 13, pages 195–200. Cambridge Univ. Press, London, 1974.