Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 140 (2005) 15–29
www.elsevier.com/locate/entcs

On-line Chain Partitioning as a Model for Real-time Scheduling

Przemyslaw Broniek[1,2]

Computer Science Department
Jagiellonian University
Krakow, Poland

Abstract

We consider on-line chain partitioning of a poset as a theoretical model for some variant of tasks scheduling. Restricting ourselves to interval orders given by its representation the problem is equivalent to the coloring problem of interval graphs. We prove that up-growing variant of on-line chain partitioning of interval orders given by its representation has an optimal solution (Nearest-Fit algorithm). We also consider on-line chain partitioning of interval orders without representation and prove the lower bound for the up-growing version $(3w/2)$. Moreover we show that there is no on-line algorithm constructing the representation of interval orders and graphs.

Keywords: on-line algorithms, scheduling, chain partitioning, up-growing, interval orders

1 Introduction

We consider a set of tasks with some relations between them. Single task has to be computed by a single processor. In a real-time situation we get one task at a time and immediately have to decide on which processor it will be computed. Our decision is irrevocable. Our objective is to minimize the number of used processors. We want to use the additional knowledge about the relations between tasks and therefore we add one rule to our scheduling. Independent tasks have to be scheduled to different processors. It is because

[1] I would like to thank my advisor Pawel Idziak for introducing me to this problem, many discussions and for his help in better understanding it.
[2] Email: broniek@ii.uj.edu.pl

doi:10.1016/j.entcs.2005.06.027

such tasks can be computed in parallel and we want to provide this for the purpose of efficiency.

This situation can also be interpreted as a two person game. The first person presents tasks and the second one assigns processors to them. A poset in which points are tasks is a perfect mathematical model for this problem. Tasks calculated by a single processor form a chain. Scheduling therefore can be considered as chain partitioning.

The off-line version of chain partitioning has a polynomial time algorithm which gives w (width of a poset) chains (Dilworth [2]). On-line (real-time) version is much more exciting. The best known on-line algorithm gives exponential number of chains (Kierstead [5], $(5^w - 1)/4$). We can also consider restricted problem, when poset is given up-growing. In this case the solution is quadratic with lower and upper bounds (Felsner [3], $w(w+1)/2$). Restricting ourselves to interval orders given by its representation the general problem is linear and equivalent to the coloring problem of interval graphs (Slusarek [8], $3w - 2$).

We prove that on-line chain partitioning of up-growing interval orders given by its representation has an optimal solution (Nearest-Fit algorithm), which is based on a greedy strategy. Considering on-line chain partitioning of interval orders without representation the situation changes dramatically. The Nearest-Fit algorithm does not work. We show that there is no on-line algorithm constructing the representation of interval orders and graphs, which forces us to design new algorithms. We also prove that the lower bound for the up-growing version of the problem changes.

In section 2 we compile necessary basic facts about posets and chain partitioning. Section 3 proceeds with the study of the on-line version of this problem and provide detailed description of our game. In the fourth section we study the class of interval orders. Following section discusses problem of chain partitioning in this class. An exact Nearest-Fit algorithm for the up-growing version is presented. Finally, last section deals with problem when the representation of interval order is not given. We prove that there is no on-line algorithm constructing such representation. A lower bound $(3w/2)$ for on-line chain partitioning of up-growing interval orders is presented.

2 Basic facts

We call a pair $\mathbf{P} = (X, P)$ *partially ordered set* (*poset* for short) if X is a set and P is a reflexive, antisymmetric and transitive binary relation on X. The set X is called *ground set* of $\mathbf{P}$ and elements of X are called *points*. Relation P is called an *order* on X.

Example 2.1 Let $X = \{1, 2, 3, 4, 5, 6\}$ and

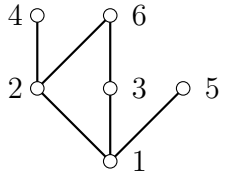
$$P = \{(x, y) \in X \times X : x \text{ divides } y\}.$$

We write $x \leq y$ and $y \geq x$ in P whenever $(x, y) \in P$. The notation $x < y$ means $x \leq y$ and $x \neq y$. Take $x, y \in P$ with $x \neq y$. We say x and y are *comparable* in P when either $x < y$ or $x > y$. Two elements are *incomparable* (denoted $x \parallel y$) when they are not comparable. We say x is *covered* by y in P, denoted $x \prec y$ in P, when $x < y$ and there is no point $z \in X$ for which $x < z < y$ in P.

We draw poset $\mathbf{P}$ on a Euclidean plane using *Hasse diagram*. This diagram is a directed acyclic graph (dag for short) $\mathbf{G} = (X, E)$ in which

$$E = \{(x, y) \in X \times X : x \prec y\}.$$

Next picture shows a diagram for the poset from example 2.1. We always assume on such diagrams that directions of edges are from the bottom to the top and do not draw them.



For a poset $\mathbf{P} = (X, P)$ and a nonempty subset Y of X, the *restriction* of P to Y, denoted by $P|_Y$, is a partial order on Y and we call $(Y, P|_Y)$ a subposet of (X, P). A poset $\mathbf{P} = (X, P)$ is called a *chain* if any two points from X are comparable in P. If $\mathbf{P}$ is a chain, we also say that P is a *linear order (total order)* on X. Similarly we call a poset an *antichain* if any two points from X are incomparable in P. A nonempty subset $Y \subseteq X$ is called a *chain* (*antichain* respectively) if the subposet $(Y, P|_Y)$ is a chain (antichain respectively). A chain C is a *maximum chain* if no other chain contains more points than C. *Maximum antichain* is defined analogously.

A point $x \in X$ is called a *maximal* point (*minimal* point respectively) if there is no point $y \in X$ such that $x < y$ in P ($x > y$ in P respectively).

The *height* of a poset (X, P) is the number of points in a maximum chain. The *width* of a poset (X, P) is the number of points in a maximum antichain.

When P and Q are partial orders on the same set X, we call Q an *extension* of P if $P \subseteq Q$, which means $x < y$ in P implies $x < y$ in Q for all $x, y \in X$.

Q is a *linear extension* of P if it is a linear order.

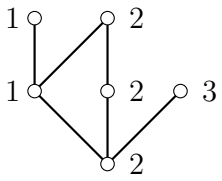
In a poset from example 2.1 the maximal points are 4, 5 and 6. Subsets $\{1,2,4\}$, $\{1,6\}$, $\{5\}$ are chains (among others). Subsets $\{4,5,6\}$, $\{2,3,5\}$, $\{4,6\}$ are antichains. Subset $A = \{3,4,5\}$ is a maximum antichain. We also have height$(\mathbf{P}) = 3$ and width$(\mathbf{P}) = 3$.

Now we can settle the problem of chain partitioning. Given $\mathbf{P} = (X, P)$ we want to find a partition of the set X into nonempty, non-intersecting chains C_i such that $X = \bigcup C_i$. Moreover, we want this partition to have minimal number of chains. We can clearly see, that there has to be at least width$(\mathbf{P})$ chains because every element of a maximum antichain must belong to different C_i. From next theorem we know, that this lower bound can always be achieved.

Theorem 2.2 (Dilworth [2]) *If* $\mathbf{P} = (X, P)$ *is a poset and* width$(\mathbf{P}) = n$, *then there exists a partition* $X = C_1 \cup C_2 \cup \ldots \cup C_n$, *where* C_i *is a chain for* $i = 1, 2, \ldots, n$.

Although it is not immediate from this theorem, we can easily get the partition into chains consisting of only non-intersecting chains. An algorithm (described for example in [7]) can compute such a partition in $O(|X|^{5/2})$ steps.

The chain partition of the poset from example 2.1 is for instance: $\{2,4\}$, $\{1,3,6\}$, $\{5\}$. We can identify chains with colors (denoted here by 1, 2 and 3) and mark points belonging to each chain with corresponding numbers as seen below:



Because of this equivalence we will sometimes identify chains with colors when dealing with on-line algorithms for chain partitioning.

Reassuming, off-line chain partitioning of a poset can be easy computed by a polynomial time algorithm and its size is equal to width of the poset.

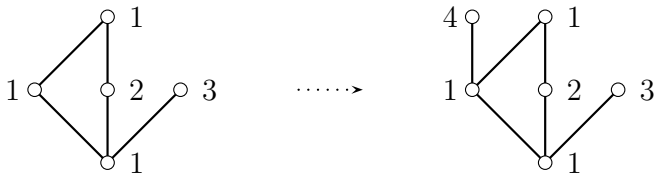
3 On-line version of the problem

On-line chain partitioning of a poset can be viewed as a game between two persons: Alice (A - the attacker) and Bob (B - the defender). The game is divided into rounds and starts with the empty poset. In each round Alice

extends the poset from the previous round with a new point x. She describes comparabilities between x and the points from previous rounds. Bob responds by assigning x to a chain. He can put it to an existing chain or create a new one. He is not allowed to move any points between chains. At each step we are interested in how many chains he created (value of the game) corresponding to how many chains are needed off-line to cover the poset, which is equal to the width of the poset. The smallest possible number of chains that Bob needs to create to cover the poset on-line is called shortly an *on-line width* of the poset.

The picture shows two consequent rounds of the game. The chains constructed by Bob are denoted by natural numbers. As mentioned earlier we call them *colors*, thus points colored with each color form a chain. New point given by the Alice is colored with 4, which means it is put into fourth chain. We can see that Bob is forced by Alice to use more colors (construct more chains) that is needed off-line to cover the poset, which we know to be 3.

Example 3.1



This example shows the essence of the game. Strategy of Alice forced defending strategy of Bob to "lose a point". It was shown by Kierstead in [6] that a greedy First-Fit strategy of Bob can be forced to use unbounded number of chains to partition a poset that have off-line width 2.

As mentioned in the introduction, on-line chain partitioning of a poset $\mathbf{P} = (X, P)$ can be used as a theoretical model for scheduling tasks. A set of tasks is the ground set X. Relations between tasks are described by an order P on them. Simply $x < y$ means that task x has to be computed before task y, for example because task y uses the results of task x. On the other hand, $x \parallel y$ means that tasks can be computed in parallel because they are independent. We can imagine that in a practical situation each task has to be calculated by a single processor. We want to schedule them by telling which processor in what order will compute which tasks. We assume that we do not know in advance the time needed to compute each task (we will relax this assumption in Section 5). This is natural because if $x < y$ and we have not computed task x then we do not know all the input, which could have effect on

time of execution, for task y yet. Moreover, we will only plan the computation and do not react later during computing in changing our schedule. This gives us one restriction to our scheduling: independent tasks have to be calculated be different processors. In a such environment, scheduling can be considered as chain partitioning. Tasks computed by each processor form a chain. We want to partition the set of tasks into smallest number of chains thus using the smallest number of processors.

In a real-time situation we get one task at a time and irrevocably assign processor to it. This situation is a practical realization of our on-line game. Alice is a spoiler of tasks and Bob is an on-line scheduler of them.

Next theorem summarizes our current knowledge about value of the chain partitioning game:

Theorem 3.2 (Szemerédi, Kierstead [5]) *For each $w \in \mathbb{N}$:*

$$\binom{w+1}{2} \leq \text{width on-line}(w) \leq \frac{5^w - 1}{4}.$$

As we can see the distance between lower and upper bounds is huge. This problem defends from attacks for over 20 years and still attracts researchers by its exponential hole. The only progress made so far is the solution of a special case:

Theorem 3.3 (Felsner [3]) width on-line$(2) = 5$.

We can say more if we change the rules of our game. The first valuable modification, given and solved by Felsner in [3], restricts legal moves of the Alice. The rule is that the sequence in which Alice gives points to Bob is a linear extension of the order. It means that in every round new point given by Alice has to be maximal. In our practical situation it means that we do not allow a new task to be computed before any one other that was given earlier. This variant of the game is called *up-growing*, because on a Hasse diagram of poset new points are always over the previous ones. In example 3.1 the move of Alice was legal according to this rule.

Our last preparatory theorem presents the solution of the problem with "up-growing" restriction, which happens to be equal to the best known lower bound of the general case:

Theorem 3.4 (Felsner [3]) *For each $w \in \mathbb{N}$:*

$$\text{width up-growing on-line}(w) = \binom{w+1}{2}.$$

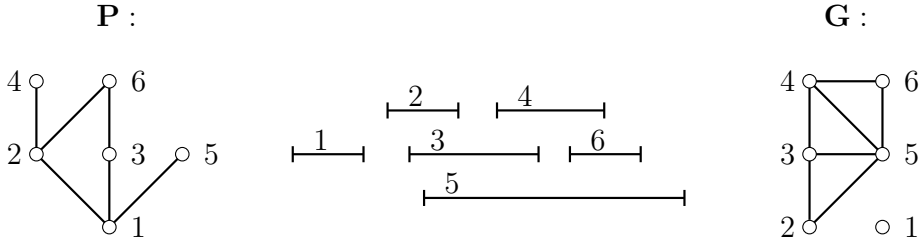
4 Interval orders

A poset is called an *interval order* whenever it can be represented with intervals on a real line. More precisely for a poset $\mathbf{P} = (X, P)$ there exists a function F assigning to each point $x \in X$ a nondegenerate closed interval $F(x) = [a_x, b_x]$ of the real line $\mathbb{R}$, so that $x < y$ in P if and only if $b_x < a_y$ in $\mathbb{R}$. The function F is called an *interval representation* of the poset $\mathbf{P}$. However, it makes no difference if degenerate or open or even unbounded intervals of $\mathbb{R}$ are taken. The key property is that of being a connected subset of $\mathbb{R}$.

An interval representation (if we concern only the intervals not a function F) can be interpreted as an interval graph.

A graph $\mathbf{G} = (V, E)$ is called an *interval graph* if there is a function F which assigns to each vertex $v \in V$ an interval (as mentioned above) $F(v) = [a_v, b_v]$, so that xy is an edge of $\mathbf{G}$ if and only if $F(x) \cap F(y) \neq \emptyset$. Similarly function F is called an *interval representation* of a graph $\mathbf{G}$.

The picture below shows common interval representation of a poset and a graph. The poset taken from example 2.1 happens to be interval.



Next theorem gives surprising characterization of interval orders:

Theorem 4.1 (Fishburn [4]) *A poset $\mathbf{P} = (X, P)$ is an interval order if and only if it does not contain $\mathbf{S_2} = \mathbf{2} + \mathbf{2}$ as a subposet.*

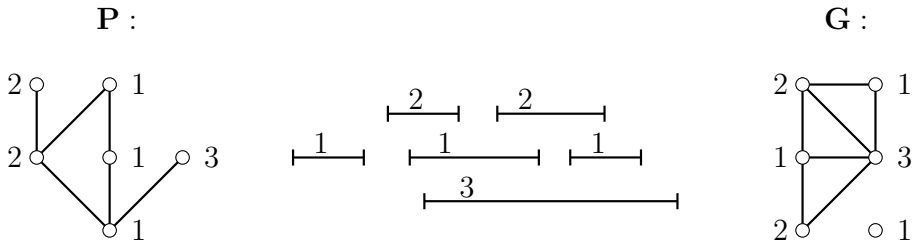
$\mathbf{S_2} = \mathbf{2} + \mathbf{2}$:

5 Nearest-Fit algorithm

After presenting interval orders, we can play our game on this class of posets. This assumes additionally, that in every round poset presented by Alice has to be an interval order. We assume that Alice gives poset by the representation. This means that we get new point x as a pair $F(x) = [a_x, b_x]$. In the next section we will see that the value of the game is different, if new points are presented without interval representation.

First note that the problem of chain partitioning for interval orders is equivalent to the problem of graph coloring for interval graphs. Indeed, a chain in an interval order is represented by a family of disjoint intervals. Such a family clearly corresponds to an anticlique in the interval graph obtained from the interval representation of the poset.

Therefore on-line coloring of an interval graph is equivalent to on-line chain partitioning of an interval order. Optimal coloring has the same number of colors as optimal chain partitioning of a corresponding poset. We can clearly see it on the picture below:



On-line coloring of interval graphs was examined much more earlier, and we know the exact lower and upper bounds for this problem:

Theorem 5.1 (Chrobak, Slusarek [1], [8]) *For each $n \in \mathbb{N}$:*

$$\text{interval graph representation coloring on-line}(n) = 3n - 2.$$

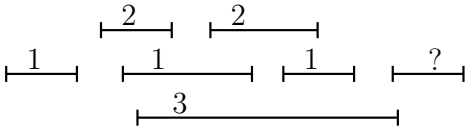
Algorithm given in [8] is even stronger. It colors broader class of *circular arc* graphs which we define in the last section. This completes the study of on-line chain partitioning of interval orders given by the representation, because we can use this algorithm and only change the interpretations of results as described.

Analyzing the up-growing version of the game we get better result, because the moves of attacker are restricted.

Theorem 5.2 *For each $w \in \mathbb{N}$:*

$$\text{interval order representation width up-growing on-line}(w) = w.$$

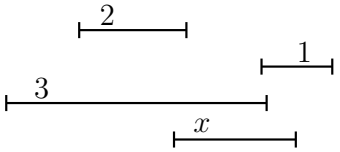
Proof. The result is achieved by using the Nearest-Fit algorithm by Bob. It is a greedy algorithm, which means it does not use new color (construct new chain) until it is forced to. Whenever there are many possibilities for a choice of a color, the algorithm apply the nearest-fit rule. It chooses such a color, which is represented by an interval with rightmost right end. This is shown on the picture below:



The algorithm chooses color number 1 from the available 1 and 2. The color number 3 is forbidden because our new interval intersects with interval colored by 3 and thus cannot form a chain. It is clear that algorithm will not work if the representation is not given. The representation makes the defense much easier, because bring more information about the poset.

We will prove the thesis by induction on w – the width of the poset after any round. If, after a round, we have $w = 1$ then the poset is a chain and we used only one color during the game. It is trivial, since every new interval was to the right of the previous one. Assuming thesis holds for every $w \leq n$ we will prove it for $w = n + 1$.

After each round the number of colors and the width increases by at most 1, because we add only one point to a poset. Let us denote the number of colors by c. Suppose, contrary to our claim, that after Bob's move we have $c > w$. It could happen only when $c = w + 1$, which means that in this particular round width of the poset has not increased and Bob was forced to use a new color. Let us analyze this situation on a picture below:

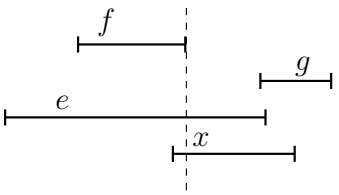


There is a new interval (denoted by x) at the bottom and some other (not all) intervals added earlier by Alice. Let us denote by E the set of the rightmost intervals from each chain constructed by Bob in the previous rounds. The number of colors including the new one is c. Thus the intervals in E are labeled from 1 to $c-1$.

Claim *Every interval $e \in E$ intersects x.*

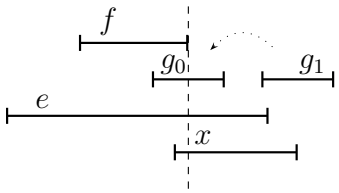
No interval can be to the right of x, because of the up-growing rule. New element has to be maximal when it is given by Alice. Suppose an interval y is on the left of x and has color z. Since it is the rightmost interval with color z, Bob could choose z for coloring x which contradicts the assumption that Bob was forced to generate a new color for x. This proves the claim.

We are going to show that there is a vertical line that intersects all together at least $w+1$ intervals. This will give a contradiction to our assumption that the width of the poset is w. Let f denote the interval from E such that the right end of f is the smallest possible among $e \in E$. Consider a dashed vertical line passing through the right end of f. It intersects f, x and maybe some other intervals from E.



Actually we will show that this dashed line has to intersect intervals of each color. Suppose that an interval $g \in E$ does not intersect the dashed line. This means that g lies entirely to the right of f. Since the poset was presented in an up-growing way, we know that g was given by Alice after f. On the other hand we know that g and f have obtained different colors, as otherwise g would represent its color in E (instead of f).

Let G be the chain of intervals that already got the same color as g, and $G_1 = \{g_1 \in G : g_1 > f\}$. Since $g \in G_1$ we can pick the smallest g_1 in G_1. If $G \setminus G_1 = \emptyset$ then g_1 is the first interval of its color. But this is impossible as there was no need for creating this color, as for example color of f could be used for g_1. Thus there is the largest element, say g_0, in $G \setminus G_1$.



Note that the left end of g_0 is not bigger that the right end of f (the dashed line), as otherwise $g_0 \in G_1$. Moreover the right end of g_0 is not smaller than the right end of f, as otherwise the Nearest-Fit strategy would propose another color for g_1, for example the color of f. Thus the dashed line intersects g_0.

This shows that each of c colors appear on intervals intersecting the dashed line. Consequently the width of the poset is a least c, contrary to our assumption that $w < c$. □

This finishes our study of chain partitioning of interval orders. In our practical application this restriction is also interesting. We only need to consider scheduling tasks in a common environment with additional time dimension. In this case we get each task with an information about start and finish time of its execution. Our Nearest-Fit algorithm gives us optimal scheduling for such presented tasks.

6 Interval orders presented without interval representation

In this section we consider the same problem of on-line chain partitioning of interval orders but this time Bob does not get the interval representation of the points Alice presents to him. Obviously our Nearest-Fit strategy does not work any longer.

We consider the up-growing version first. In this case Alice can force Bob to construct more chains presenting him an interval order:

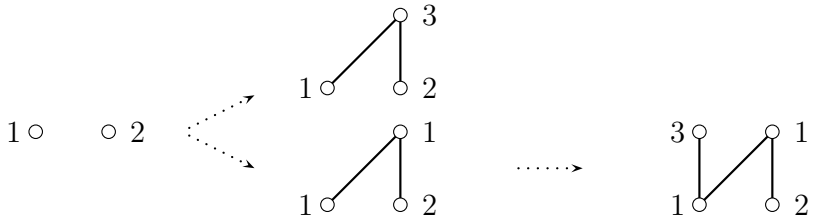
Theorem 6.1 *For each $w \in \mathbb{N}$:*

$$\text{interval order width up-growing on-line}(w) \geq \frac{3w}{2}.$$

Proof. Let us first show that:

$$\text{interval order width up-growing on-line}(2) \geq 3.$$

This is shown on the picture below:

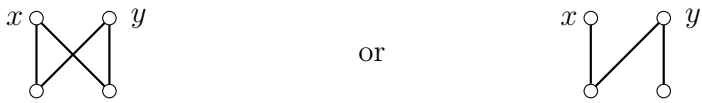


Alice presents first two incomparable elements and Bob is forced to use two different colors, say 1 and 2, for these points. Now Alice adds a new point that dominates both of the two old ones. Bob either uses the third color for it (so we are done), or uses one of the old colors. Without loss of generality this old color could be chosen to be 1. Then Alice add the fourth point that dominates only the one of the minimal points that was colored by 1. This obviously forces Bob to use the third color at this point. It is easy to check that the width of the presented poset is 2, and it is interval.

This construction can be extended to get our theorem as follows. First an antichain $A = \{a_1, \ldots, a_w\}$ of $w = 2n$ elements is presented to Bob, so that he is forced to use w different colors on them. Then Alice replies with an antichain of n new points $B = \{b_1, \ldots, b_n\}$ that dominates all points in A. Bob can color B either by using new colors or the old ones. Let k be the number of old colors used for B, and let $A_1 = \{a_i \in A :$ there is $b_j \in B$ with the same color as $a_i\}$. Since $|A_1| = k \leq n$ there is an n-element set A_1' such that $A_1 \subseteq A_1' \subseteq A$. This makes it possible for Alice to present an antichain of n new points $C = \{c_1, \ldots, c_n\}$ such that $x < c_i$ iff $x \in A_1'$.

Obviously the colors used for $A \setminus A_1'$ cannot be used for C. But also the colors of A_1 can not be used for C as they are already used on B. This forces Bob to use at least k new colors on C. Together with the fact that $n-k$ colors not present at A were used on B, this gives that at least $w + (n-k) + k = 3n = 3w/2$ colors were used by Bob.

Now the Theorem follows from the easily observed fact that the width of the poset is w and that the poset is interval. The last claim can be check by using Theorem 4.1: for two maximal points x, y the downset of $\{x, y\}$ is:



□

Our Nearest-Fit algorithm needs an interval representation of a poset (graph) to work. If we can construct (on-line) the representation it needs, we could apply first this on-line representation and then the Nearest-Fit algorithm. Now we consider constructing representation for a poset as an on-line game. Alice presents poset to Bob point by point, but Bob have to construct the interval representation of it. In view of theorem 6.1 there is no algorithm constructing interval representation of an interval order, as otherwise it would give enough information for Bob to do optimal chain partitioning using Nearest-Fit strategy.

Next we consider the general case (not necessarily up-growing). As we have already mention this is equivalent to coloring of interval graphs. Theorem 5.1 solves the problem if the interval graph is presented together with its interval representation. But can such an interval representation be obtained on-line? The answer is negative. We prove even stronger result, for which we need the following definition.

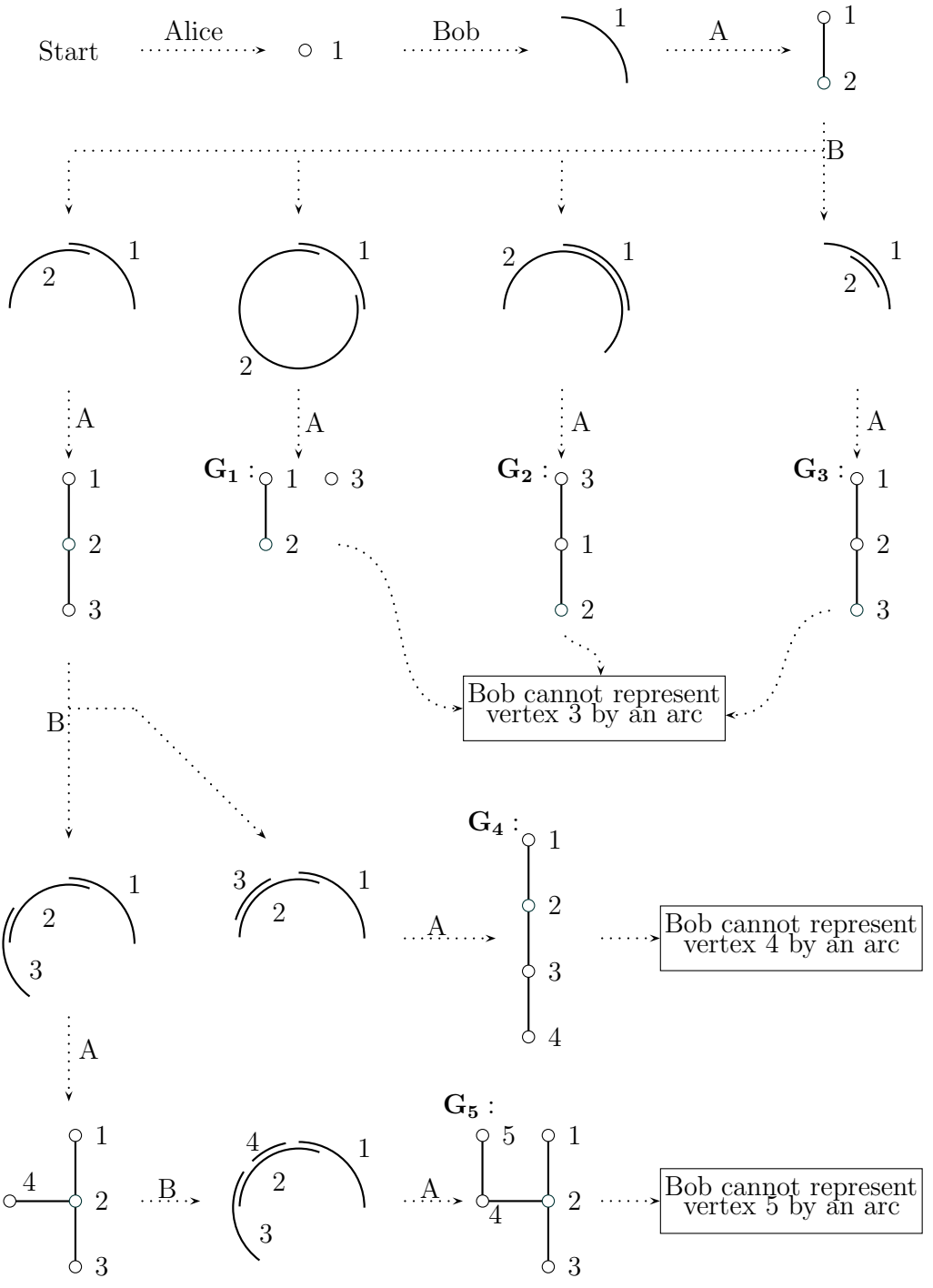
A graph $\mathbf{G} = (V, E)$ is called a *circular arc graph* if there is a function F which assigns to each vertex $v \in V$ a nondegenerate arc $F(v)$ of a circle, so that xy is an edge of $\mathbf{G}$ if and only if $F(x) \cap F(y) \neq \emptyset$. The function F is called a *circular arc representation* of the graph $\mathbf{G}$.

The circle gives us more possibilities for representing interval graphs. Every interval graph is a circular arc graph. On the other hand, circular arc graph happens to be interval when in one of its representations on a circle there is an uncovered point. We can do a cut in this point and get an interval representation.

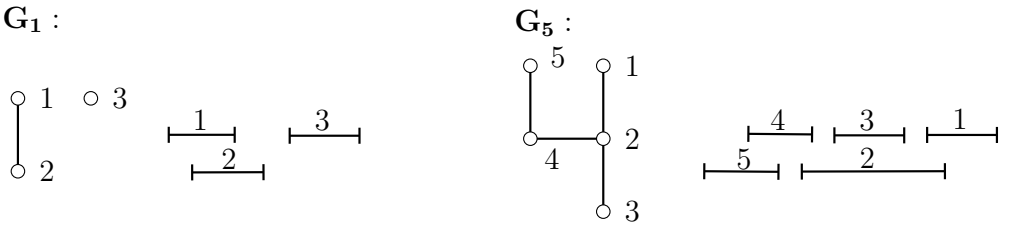
Theorem 6.2 *There is no on-line algorithm constructing circular arc representation for the interval graph.*

Proof.

The picture below shows the full tree of a game in which Alice presents vertices of a graph and Bob responds by constructing the circular arc representation. The representations constructed by Bob are presented up to isomorphism without the loss of possibilities of his moves:


Start
Alice
Bob
A
B
G_1 :
G_2 :
G_3 :
G_4 :
G_5 :
Bob cannot represent vertex 3 by an arc
Bob cannot represent vertex 4 by an arc
Bob cannot represent vertex 5 by an arc

Inspecting this tree we easy note that Bob looses as he cannot represent the last point added by Alice. On the other hand all graphs produced by Alice are not only circular arc graphs but in fact they are interval graphs. Paths G_2, G_3 and G_4 are trivially interval graphs while G_1 and G_5 have interval (and therefore circular arc) representations presented below:



□

Theorem 6.2 tells us that we need to design new algorithm for chain partitioning problem of interval posets when they are presented without their interval representation. We strongly believe that chain partitioning of interval orders without representation has an on-line linear algorithm for both upgrowing and general version. We know only a lower bound, so this is an open problem for now.

References

[1] M. Chrobak, M. Slusarek, *On some packaging problem related to dynamic storage allocation*, Theoretical Informatics and Applications **22** (1988), 487–499.

[2] R. P. Dilworth, *A Decomposition Theorem for Partially Ordered Sets*, Ann. Math. **51** (1950), 161–165.

[3] S. Felsner, *On–line chain partitions of orders*, Theoret. Computer Science **175** (1997), 283–292.

[4] P. C. Fishburn, *Intransitive Indifference with Unequal Indifference Intervals*, J. Math. Psych. **7** (1970), 144–149.

[5] H. A. Kierstead, *An effective version of Dilworth theorem*, Trans. Amer. Math. Soc. **268** (1981), 63–77.

[6] H. A. Kierstead, *Recursive ordered sets*, Contemp. Math. **57** (1986), 75–102.

[7] K. Kloch, *Off–line and on–line dimension of partial orders*, Master Thesis, Jagiellonian University (2003).

[8] M. Slusarek, *Optimal on–line coloring of circular arc graphs*, Theoretical Informatics and Applications **29** (1995), 423–429.

[9] W. T. Trotter, *Combinatorics and partially ordered sets: dimension theory*, John Hopkins Press, 1992.