Discrete Mathematics and Theoretical Computer Science DMTCS vol. **22**:1, 2020, #17

On the Heapability of Finite Partial Orders

János Balogh[*1] Cosmin Bonchiş[†2,3] Diana Diniş[2,3] Gabriel Istrate[‡2,3] Ioan Todinca[§4]

[1] *Institute of Informatics, University of Szeged, Hungary*
[2] *Department of Computer Science, West University of Timişoara, Timişoara, Romania.*
[3] *e-Austria Research Institute, Timişoara, Romania*
[4] *LIFO, Université d'Orléans and INSA Centre-Val de Loire, France*

received 16th May 2018, *revised 25th Mar. 2020*, *accepted 12th May 2020*.

Define a sequence of elements from a partially ordered set to be *heapable* if it can be successively inserted as the leaves of a binary tree, not necessarily complete, such that children nodes are always greater or equal than parent nodes. This is a natural binary analog of the notion of a chain in a poset and an easy extension of a definition for integers due to Byers et al. (2011). A set of elements is called heapable if some permutation of its elements is.

We investigate the partitioning of sequences from a poset into a minimal number of heapable subsequences. We give an extension of Fulkerson's proof of Dilworth's theorem to decomposition into heapable subsequences which yields as a byproduct a flow-based algorithm for computing such a minimal decomposition.

On the other hand, for sets and sequences of intervals and for trapezoid partial orders we prove that such minimal decompositions can be computed via simple greedy-type algorithms.

Second, while the complexity of computing a maximal heapable subsequence of integers is still open, we show that this problem has a polynomial time algorithm for sequences of *intervals*.

The paper concludes with a couple of open problems related to the analog of the Ulam-Hammersley problem for sets and sequences of random intervals.

Keywords: partial order, heapable sequence, Dilworth's theorem, greedy algorithm, random intervals.

[*]Supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

[†]Supported in part by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, project no. PN-II-RU-TE-2014-4-0270 - FraDys: "Theoretical and Numerical Analysis of Fractional-Order Dynamical Systems and Applications".

[‡]Corresponding author: Gabriel Istrate. G. I. and D.D. were supported in part by a grant of Ministry of Research and Innovation, CNCS - UEFISCDI, project number PN-III-P4-ID-PCE-2016-0842, within PNCDI III, ATCO: "Advanced techniques in optimization and computational complexity".

[§]Partially supported by the ANR (Agence Nationale pour la Recherche) project GraphEn, ANR-15-CE40-0009.

ISSN 1365–8050

1 Introduction

The *longest increasing subsequence* is a classical problem in combinatorics and algorithmics. Decompositions of (random) permutations into a minimal number of increasing sequences have been studied in the context of the famous *Ulam-Hammesley problem*. This problem concerns the asymptotic scaling behavior of the length of the longest increasing subsequence (LIS) of a random permutation, and is a problem with deep connections to statistical physics and random matrix theory (for a very readable introduction see Romik (2015)).

An interesting variation on the concept of increasing sequence was introduced by Byers et al. Byers et al. (2011): call sequence of integers $A = a_1, a_2, \ldots, a_n$ *heapable* if numbers $a_1, a_2, \ldots, a_n$ can be successively inserted as leaves into a heap-ordered binary tree (not necessarily complete). Heapability is a "weakly increasing pattern" for integer sequences. The study of patterns in (random) integer sequences (Crane and DeSalvo (2018)), has been, of course, quite popular lately, some of the investigated problems even involving increasingly labelled trees (Durant and Wagner (2019)) or heaps (Defant (2019)).

Heapability was further investigated in Istrate and Bonchiş (2015) (and, independently, in Porfilio (2015)). In particular, a subgroup of the authors of the present papers showed that for permutations one can compute in polynomial time a minimal decomposition into heapable subsequences, and investigated a problem that can be viewed as an analog of the Ulam-Hammersley problem for heapable sequences (see also subsequent work in Istrate and Bonchiş (2016); Basdevant et al. (2016); Basdevant and Singh (2018); Bonchiş et al. (2018); Chandrasekaran et al. (2019), that extends/confirms some of the conjectures of Istrate and Bonchiş (2015)). Furthermore, as shown in Istrate and Bonchiş (2016) one can meaningfully study the analogs of heapability and the Ulam-Hammesley problem in the context of *partial orders*.

This paper started as a conversation on the heapability of *sequences of intervals* during a joint Timişoara Szeged seminar on theoretical computer science in November 2015. Its main purpose is *to offer a different perspective on the concept of heapability, by relating it to well-known results in combinatorics, such as the classical theorems of Dilworth and Kőnig-Egerváry.*

Specifically, we prove that the number of classes in a minimal decomposition of a sequence of elements from a poset P into "heapable subsequences" can be obtained as the size of a minimum vertex cover in a certain bipartite graph whose construction directly generalizes the one employed in one classical proof of Dilworth's theorem. As a byproduct, we obtain an efficient algorithm based on network flows for computing such a minimal decomposition.

This result, together with the ones from Istrate and Bonchiş (2015) (where such a minimal decomposition was computed, for integer sequences, via a direct, greedy algorithm) raise the question whether such greedy algorithms exist for other posets except the set of integers. We answer this question in the affirmative way by showing that the result of Istrate and Bonchiş (2015) is extendible to *sets and sequences of intervals*, ordered by the natural partial order.

The structure of the paper is as follows: in Section 2 we review the main concepts and technical results we will be concerned with. Then, in Section 3 we prove a result (Theorem 1) which shows that part of Fulkerson's proof of Dilworth's theorem (Fulkerson (1956)) can be extended to characterizing partitions into a minimal number of heapable subsequences. This result provides, as a byproduct, a flow-based algorithm for the computation of such a minimal partition.

We then investigate, in Section 4, the heapability of sequences of intervals, showing (Theorem 2) that a greedy-type algorithm computes such a minimal partition. In Section 5 we show (Theorem 3) that this result extends to (unordered) *sets* of intervals as well. We then give in Section 6 a second, incomparable,

extension (Theorem 4) from sequences of intervals to sequences from so-called *trapezoid* partial orders.

In Section 7 we investigate the problem of computing a maximal heapable subsequence of a sequence of elements from a poset. For sequences of integers the complexity of this problem is open (Byers et al. (2011)). We show (Theorem 5) that it has polynomial time algorithms for sequences of intervals.

We conclude in Section 8 with some open questions raised by our results.

2 Preliminaries

We will assume knowledge of standard graph-theoretic notions. In particular, given integer $k \geq 1$, rooted tree T is *k-ary* if every node has at most k children. Given a graph $G = (V, E)$, we will denote by $vc(G)$ the size of the minimum vertex cover of G.

Definition 1. *Let $k \geq 1$. A sequence of integers $A = (a_1, a_2, \ldots, a_n)$ is called* k-heapable *(or a k-ary* chain*) if there exists a k-ary tree T with n nodes labeled by $a_1, a_2, \ldots, a_n$, such that for any two nodes labeled a_i, a_j, if a_j is a descendant of a_i in T then $i < j$ holds for their indices and $a_i < a_j$.*

We will be concerned with finite partially ordered sets (posets) only. Sequences of elements from a poset naturally embed into this framework by associating, to every sequence $A = (a_1, \ldots, a_n)$ the poset $Q_A = \{(i, a_i) : 1 \leq i \leq n\}$ with partial order $(i, a_i) \leq (j, a_j)$ if and only if $i \leq j$ and $a_i \leq a_j$. Given poset Q, its subset B is a *chain* if the partial order of Q is a total order on B. Subset C is an *antichain* if no two elements a, b of C are comparable with respect to the partial order relation of Q. Dilworth's theorem (Dilworth (1950)) states that the minimum number of classes in a chain decomposition of a partial order Q is equal to the size of the largest antichain of Q.

One particular type of posets we consider is that of *permutation orders*: $\preceq$ is a *permutation order* if there exists a permutation π of the elements of U such that, for all $a, b \in U$, $a \preceq b$ iff $a < b$ and $\pi^{-1}(a) < \pi^{-1}(b)$.

Another particular case we will be concerned with is that of *interval orders*. Without loss of generality all our intervals will be closed subsets of $(0, 1)$. We define a partial order of them as follows: Given intervals $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ with $a_1 < b_1$ and $a_2 < b_2$, we say that $I_1 \leq I_2$ if and only if the entire interval I_1 lies to the left of I_2 on the real numbers axis, that is $b_1 \leq a_2$. For technical reasons we will also require a total ordering of intervals, denoted by $\sqsubseteq$ and defined as follows: $I_1 \sqsubseteq I_2$ if either $b_1 < b_2$ or $b_1 = b_2$ and $a_1 < a_2$.

We will consider in this paper models of random intervals. Similarly to the model in Justicz et al. (1990) by "random intervals" we will mean random subintervals of $(0, 1)$ generated as follows: A random sample I can be constructed iteratively at each step by choosing two random real numbers $a, b \in (0, 1)$ and taking $I = [min(a, b), max(a, b)]$.

The *patience sorting algorithm* (Mallows (1963)) partitions a permutation into a minimal number of increasing subsequences of integers. It works by adding each element in an online fashion to the first subsequence where it can be added, starting a new subsequence if no existing one is compatible.

We need to briefly review one classical proof of Dilworth's theorem, due to Fulkerson (Fulkerson (1956)): First, given poset $Q = (U, \leq)$ with n elements, define the so-called *split graph associated to* Q (Felsner et al. (2003)), to be the bipartite graph $G_Q = (V_1, V_2, E)$, where $V_1 = \{x^- : x \in U\}$ and $V_2 = \{y^+ : y \in U\}$ are independent copies of U, and given $x < y \in U$ we add to E edge $x^- y^+$. Fulkerson proved that each chain decomposition of Q uniquely corresponds to a matching in G_Q. The proof proceeded by employing the classic Kőnig-Egerváry theorem (Konig (1931); Egerváry (1931)),

stating that the size of a maximum matching in a bipartite graph is equal to the minimum vertex cover in the same graph. This result was applied to the bipartite graph G_Q, inferring that

Proposition 1. *The cardinality of a minimum chain decomposition of Q is equal to $n - vc(G_Q)$, where $vc(G_Q)$ is the vertex cover number of the split graph G_Q.*

Finally, Fulkerson's proof of Dilworth's theorem concluded by showing that $n - vc(G_Q)$ is equal to the size of the largest antichain of Q.

We extend the definition of k-heapable sequences to general posets as follows:

Definition 2. *Given integer $k \geq 1$ and poset $Q = (U, \leq)$ a subset A of the ground set U is called k-heapable (or, equivalently, a k-ary chain of Q) if there exists a k-ary rooted tree T and a bijection between A and the vertices of T such that for every $i, j \in A$, if j is a descendant of i in T then $i < j$ in Q.*

We stress the fact that the notion of a k-ary chain above is distinct from the notion of k-chain that appears in the statement of the Greene-Kleitman theorem (Greene and Kleitman (1976)).

Definition 3. *The k-width of poset Q, denoted by k-wd(Q), was defined in Istrate and Bonchiş (2016) as the smallest number of classes in a partition of Q into k-ary chains. For $k = 1$, by Dilworth's theorem, we recover the usual definition of poset width (Trotter (1995)).*

Observation 1. *In the previous definition we partition a* set *into k-ary chains. On the other hand in the problem studied in Istrate and Bonchiş (2015) we partition a permutation (i.e.* sequence *of elements) into k-heapable subsets.*

There is no contradiction between these two settings, as one can map a permutation $\pi \in S_n$ of n elements onto a poset defined as $\{(i, \pi[i]) : i = 1, \ldots, n\}$ with

$$(i, \pi[i]) < (j, \pi[j]) \text{ iff } i < j \text{ and } \pi[i] < \pi[j]$$

Two important (unpublished) minimax theorems, attributed in Gyárfás and Lehel (1985) to Tibor Gallai and ultimately subsumed by the statement that interval graphs are perfect, deal with sets of intervals. We will only be concerned with the first of them, that states that, given a set of intervals J on the real numbers line the following equality holds:

Proposition 2. *The minimum number of partition classes of J into pairwise disjoint intervals is equal to the maximum number of pairwise intersecting intervals in J.*

Of course, the first quantity in Proposition 2 is nothing but the 1-width of the partial order $\leq$ on intervals. The scaling of (the expected value of) this parameter for sets R of n random intervals has the form (Justicz et al. (1990))

$$E[\text{1-wd(R)}] = \frac{2}{\sqrt{\pi}}\sqrt{n}(1 + o(1)).$$

2.1 A graph-theoretic interpretation

Problems that we are concerned with have, it turns out, an algorithmic interpretation that is strongly related to problems of computing the maximum independent sets and the chromatic number of various classes of perfect graphs. The idea first appeared (somewhat implicitly) in Porfilio (2015). In this subsection we make it fully explicit as follows:

Definition 4. *Given a directed graph $G = (U, E)$, call a set of elements $W \subseteq U$ a* k-treelike independent set of G *if there is a rooted k-ary tree T, not necessarily complete, and a bijection $f : V(T) \to W$ such that, for all vertices v, w of T, if v is an ancestor of w in T then $f(v)$ and $f(w)$ are* **not** *connected by an edge in G.*

A poset can, of course, be viewed as a directed graph, so the previous definition applies to posets as well. We can apply the concept to undirected *graphs as well by identifying such a graph with its oriented version containing, for any undirected edge e, both directed versions of e.*

For undirected graphs a 1-treelike independent set of G is simply an independent set: indeed, the condition in the definition simply enforces the nonexistence of edges between the vertices in W. Paralleling the case $k = 1$ (for which a polynomial time algorithm is known, in the form of patience sorting), we can restate the open problem from Byers et al. (2011) as the problem of computing a maximum 2-treelike independent set:

Proposition 3. *Computing the longest k-heapable subsequence of an arbitrary permutation (order) π is equivalent to computing a maximum k-treelike independent sets in the digraph induced by the transitive closure of the Hasse diagram of the associated partial order P_π.*

When k is implied, we will informally use the name *maximum heapable subset* for the problem of computing a maximum k-ary chain of an arbitrary permutation. Similarly, the problem of partitioning the set into k-ary chains can be regarded as a generalization of graph coloring: define a *k-treelike coloring of a partial order P* to be a partition of the universe U into classes that induce k-treelike independent sets. The *k-treelike chromatic number* of a partial order P is the minimum number of colors in a k-treelike coloring of P.

The main result from Istrate and Bonchiş (2015) is equivalent to the following (re)statement:

Proposition 4. *For $k \geq 2$ the greedy algorithm of Figure 1 computes an optimal k-treelike coloring of permutation partial orders.*

Input: Permutation order P specified by permutation $\pi \in S_n$.
Output: A k-treelike coloring of P.

let $F = \emptyset$
for $i := 1$ to n do:

if $\pi(i)$ cannot become the child of any node in F.

then

start a new tree with root $\pi(i)$.

else

make $\pi(i)$ a child of the largest value $\pi(j) < \pi(i)$,
$j < i$ that has fewer than k children.

Fig. 1: The greedy best-fit algorithm for k-tree-like colorings of permutation posets.

Again, let us remark that the corresponding statement for $k = 1$ is well known, as it is equivalent to the greedy coloring of permutation graphs, accomplished by patience sorting.

3 Main result

Our main result proves a k-ary extension of Proposition 1. To state it, we extend the definition of split graphs to all values $k \geq 1$ as follows:

Definition 5. *Given poset $Q = (U, \leq)$ an integer $k \geq 1$, the k-split graph associated to poset Q is the bipartite graph $G_{Q,k} = (V_1, V_2, E)$, where*

- $V_1 = \{x_1^-, \ldots, x_k^- : x \in U\}$
- $V_2 = \{y^+ : y \in U\}$
- *given $x, y \in U$, $x < y$, add to E edge $x_i^- y^+$ for $i \in 1, ..., k$.*

Given this definition, our main result shows that computing the k-width of a finite poset can be computed as in Fulkerson's proof of Dilworth's theorem, by directly generalizing Proposition 1:

Theorem 1. *Let $Q = (U, \leq)$ be a finite poset with n elements and a fixed integer $k \geq 1$. Then*

$$k\text{-}wd(Q) = n - vc(G_{Q,k}). \tag{1}$$

Proof: Define a *left k-matching* of graph G_Q to be any set of edges $A \subseteq E$ such that for every $x, y \in U$, $deg_A(x^-) \leq k$ and $deg_A(y^+) \leq 1$.

Claim 1. *Partitions of Q into k-ary chains bijectively correspond to left k-matchings of G_Q. The number of classes of a partition is equal to n minus the number of edges in the associated left k-matching.*

Proof: Consider a left k-matching A in G_Q. Define the partition P_A as follows: roots of the k-ary chains consist of those $x \in U$ for which $deg_A(x^+) = 0$. There must be some element $x \in U$ satisfying this condition, as the minimal elements of Q with respect to $\leq$ satisfy this condition.

Now we recursively add elements of U to the partition P_A (in parallel) as follows:

1. All elements $y \in U$ not yet added to any k-ary chain, and such that y^+ is connected to some x^- by an edge in A are added to the k-ary chain containing x, as direct descendants of x. Note that element x (if there exists at least one y with this property) is unique (since $deg_A(y^+) = 1$ in this case), so the specification of the k-ary chain to add y to is well defined. On the other hand, each operation adds at most k successors of any x to its k-ary chain, since $deg_A(x^-) \leq k$.

2. If all direct predecessors of an element $x \in U$ have been added to some k-ary chain and are no longer leaves of that k-ary chain then x will be the root of a new k-ary chain.

Conversely, given any partition P of U into k-ary chains, define set of edges A consisting of edges x^-, y^+ such that x is the parent of y in a k-ary chain. It is immediate that A is a left k-matching. □

Corollary 1. *There is a bijective mapping between partitions of Q into k-ary chains and matchings of $G_{Q,k}$ such that the number of k-ary chains in a partition is n minus the number of edges in the matching.*

Proof: We will actually show (using Claim 1) how to associate left k-matchings of G_Q to matchings of $G_{Q,k}$. The idea is simple: given a node x^- of G_Q with $l \leq k$ neighbors in V_2, construct a matching in $G_{Q,k}$ by giving each of $x_1^-, \ldots, x_l^-$ exactly one neighbor from the neighbors of x^- (in a pairwise distinct way). In the other direction, if $e = x_i^- y^+$ is an edge in the matching of $G_{Q,k}$ then consider the appropriate edge $x^- y^+$ in G_Q. It is easy to check that it gives a left k-matching in G_Q (which corresponds to a $k-$chain partition of Q). Furthermore, the number of $k-$chains is the number of edges in the left $k-$matching in G_Q which is the same as n minus $vc(G_Q)$. □

We complete the proof of Theorem 1 (based on Claim 1 and Corollary 1) by applying in a straightforward way Kőnig's theorem to the graph $G_{Q,k}$. □

Corollary 2. *One can compute parameter k-wd(Q) by creating a flow network Z_Q and computing the value of the maximum flow of Z_Q consisting of :*

- *vertices and edges of G_Q, with edge capacity 1.*
- *a source s, connected to nodes in V_1 by directed edges of capacity k,*
- *a sink t, that all nodes in V_2 connect to via oriented edges of capacity* 1.

computing the maximum s-t network flow value f in network Z_Q and outputting k-wd(Q)=n-f.

Proof: Straightforward, this is simply the maximal flow algorithm for computing the maximal size left k-matching in G_Q, similar to the construction for maximum matchings in bipartite graphs in the literature. □

4 Heapability of sequences of intervals: an online algorithm

We now know that the problem of computing a minimal partition of the elements of a poset into heapable subsequences has a polynomial time algorithm.

On the other hand for permutations (sequences of integers) the optimal algorithm presented in Istrate and Bonchiş (2015), which extended the well-known *patience sorting algorithm* (Mallows (1963)), had a simple, online, structure: the heaps were built incrementally, by considering the elements one by one and inserting each element as a leaf in one of the existing heaps, or starting a new heap. In contrast, the particularization of the network flow algorithm to permutations is **not** online in any natural sense, as it computes a maximum flow in a graph which depends "globally" on the sequence. One could ask whether the existence of the online algorithm from Istrate and Bonchiş (2015) is an exception or, rather, such online algorithms which are optimal exist for sequences of elements from other posets as well.

The question is also interesting since the case $k = 1$ is a natural variant of the *online chain partitioning problem*, a problem with a rich history (Kierstead et al. (1984); Trotter (1995)) and bibliography (see e.g. Bosek et al. (2012)). In contrast to the classical case, in our variant elements can be assigned to a (k-ary) chain only if they can be inserted *as leaves* at the moment they are inserted.

In the sequel we provide an affirmative answer to the above question, by focusing on the case of sequences of intervals:

Theorem 2. *For every fixed $k \geq 1$ there exists a (polynomial time) online algorithm that, given a sequence of intervals $S = (I_1, I_2, ..., I_n)$ as input, computes a minimal partition of S into k-ary chains so that each interval is inserted at the time of its consideration as a leaf into one of the k-ary chains, or starts a new k-ary chain.*

Before proceeding with the proof of theorem 2, let us remark a potential application of a variant of this result to parallel computing: many algorithms in this area (e.g. algorithms using a *parallel prefix-sum* design methodology, Blelloch (1990)) require the computation of all prefixes of an associative operation $A_1 * A_2 * \ldots A_n$. Operations being performed (each corresponding to one computation of a *-product) are arranged on a binary tree. In the (completely equivalent) max-heap variant of theorem 2, children intervals are required to be less or equal to the parent interval with respect to ordering $\leq$. This is quite natural from the standpoint of parallel computing: consider the setting of a parallel-prefix problem where each intermediate *-computation is a rather costly operation; intervals now represent times when these operations can be scheduled. The requirement that the parent interval be larger than child intervals with respect to $\leq$ is completely natural, as child computations need to complete in order to feed their results to the parent computation. Thus our result answers the question whether all the time intervals can be scheduled on a single heap-ordered binary tree, and gives such a scheduling, if the answer is affirmative.

4.1 Proof of Theorem 2

The proof employs the concept of *slots*, adapted for interval sequences from similar concepts for permutations (Byers et al. (2011); Istrate and Bonchiş (2015)):

Definition 6. *When a new interval is added to a k-ary chain it opens k new positions to possibly insert other intervals as direct successors into this node. Each position has an associated integer value that will be called its* slot*. The value of all empty slots created by inserting $I_1 = [a_1, b_1]$ into a k-ary chain will be b_1, the right endpoint of I_1.*

Definition 7. *An interval I is compatible with an (empty) slot with value x if all of I lies in $[x, \infty)$.*

Intuitively, x is the smallest value of the left endpoint of an interval that can be inserted in the k-ary chain as a child of an interval I_1 with right endpoint x while respecting the heap property. Indeed, as k-ary chains are (min-)heap ordered, an insertion of an interval I into a k-ary chain as a child of I_1 is legal if the interval I is greater than I_1 with respect to $\leq$ relation. This readily translates to the stated condition, that the start point of I must be greater or equal than the slot value of its parent.

The proposed greedy best-fit algorithm for computing a minimum partition into k-chains a sequence of n intervals is described in Figure 2. As an example, consider the sequence of intervals S_1 below, with $k = 2$. The resulting configuration is shown in Figure 3.

$$S_1 = ([1,7],[1,11],[11,12],[15,16],[7,9],[8,16],[1,2],[3,19],[13,17],[5,7])$$

By choosing the highest valued slot available for insertion for $I_t = [a_t, b_t]$, we make sure that the difference between the chosen slot value s and a_t is minimal. This is desirable because there may be some interval further down the sequence with starting point value in between the s and a_t that fits slot s but cannot be inserted there, as the slot is no longer available.

We define the concepts of signature of a multiset of slots and domination between such multisets in a similar way to the corresponding concepts for permutations in Byers et al. (2011):

Input: A sequence of intervals $I = (I_1, I_2, \ldots, I_n)$.
Output: A partition H of I into k-ary chains.

for $i := 1$ to n do:

if $I_i = [l_i, r_i]$ can be inserted into some empty slot

then insert I_i in the highest-valued compatible slot (a child of the node with this slot).

else create a new k-chain rooted at I_i

Fig. 2: The best-fit algorithm for k-ary chain partition of sequences of intervals.

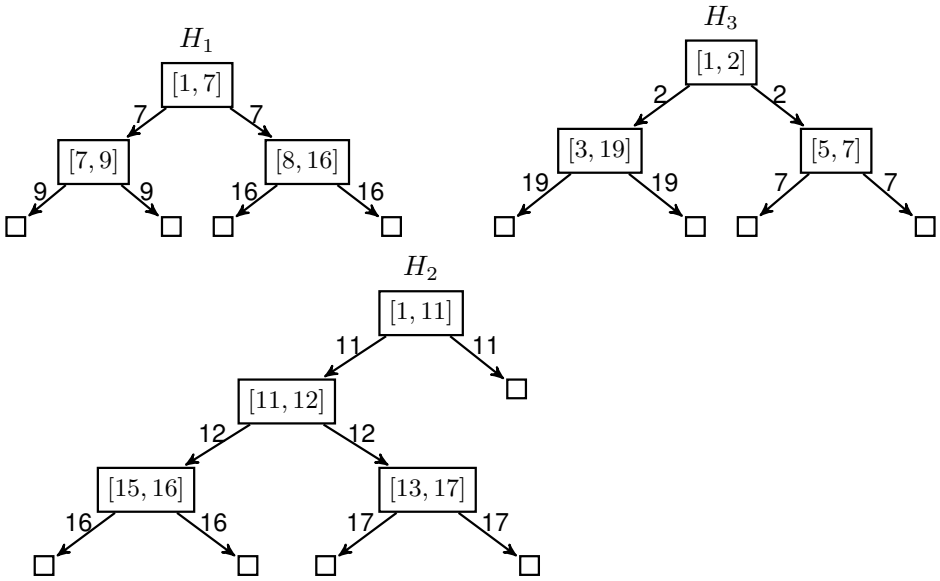

Fig. 3: The binary (2-ary)-chain configuration corresponding to S_1.

Definition 8. *Given a multiset of slots H, we call the vector of slots, sorted in increasing order, the* signature *of H. We shall denote this by $sig(H)$. By slightly abusing notation, we will employ the previous definition in the obvious way when H is a union of k-chains as well.*

Definition 9. *Multiset P dominates multiset Q (denoted $P \preceq Q$) if $|sig(P)| \leq |sig(Q)|$ and for all $1 \leq i \leq |sig(P)|$ we have $sig(P)[i] \leq sig(Q)[i]$.*

For example, for the binary (2-ary) chains in the Figure 3 their corresponding signatures are, respectively:

$sig(H_1) = [9, 9, 16, 16]$;

$sig(H_2) = [11, 16, 16, 17, 17]$;

$sig(H_3) = [7, 7, 19, 19]$.

Therefore, in our example $H_1 \preceq H_2$, (H_1 dominates H_2), but no other domination relations between H_1, H_2, H_3 are true.

Lemma 1. *Assume that A, B are multisets of slots and A dominates B. Then the multisets A' and B' obtained after inserting a new interval into the largest compatible slot of A, and into an arbitrary compatible slot of B, propagates the domination property, i.e. A' dominates B'.*

Proof:

Let $sig(A) = [a_1, a_2, \ldots, a_{|sig(A)|}]$ and $sig(B) = [b_1, b_2, \ldots, b_{|sig(B)|}]$. Also, by convention, define $a_0 = b_0 = -\infty$ and $a_{|sig(A)|+1} = b_{|sig(B)|+1} = +\infty$.

Proving that $A' \preceq B'$ is equivalent to proving that $|sig(A')| \leq |sig(B')|$ and for all indices $1 \leq l \leq |sig(A')|$:

$$sig(A')[l] \leq sig(B')[l]. \tag{2}$$

The cardinality condition can be easily verified: indeed, $|sig(A')|$ is either $|sig(A)| + k - 1$ (if the process adds k copies of y but also kills a lifeline) or $|sig(A)| + k$ (no slot exists with a value less or equal to x), and similarly for $|sig(B')|$. It follows easily from the domination property that when $|sig(A')| = |sig(A)| + k$ then $|sig(B')| = |sig(B)| + k$ as well. Indeed, since $|sig(A')| = |sig(A)| + k$, no slot of A has value lower than x (or it would lose a lifeline). Thus $a_1 > x$ and since $A \preceq B$, $b_1 \geq a_1 > x$. So no slot of B is smaller than x either.

As for the second condition, consider the slots from A and B which interfere with the newly arrived interval as follows:

- Let i and i' be the (unique) indices such that $a_i \leq x < a_{i+1}$ and $a_{i'} \leq y < a_{i'+1}$ hold in A, with $i, i' \in \{0, \cdots, |sig(A)|\}$.
- Similarly, let j and j' be the unique indices such that $b_j \leq x < b_{j+1}$ and $b_{j'} \leq y < b_{j'+1}$ hold in B, with $j, j' \in \{0, \cdots, |sig(B)|\}$.

Since $A \preceq B$, it follows that $i \geq j$ and $i' \geq j'$. Suppose that we insert the interval $[x, y]$ in B in an arbitrary slot $b_t \leq b_j$ (thus removing one life of slot b_t and inserting k copies of slot y). The rest of the proof is by a case analysis. The four cases which can be distinguished (displayed in Fig. 4, for $k = 2$) are:

Case 1. $l < t$:

In this case, none of the signatures A, B were affected at position l by the insertion of $[x, y]$, hence:

$$sig(A')[l] = sig(A)[l] \leq sig(B)[l] = sig(B')[l].$$

Case 2. $l \in [t, j')$:

In this range all slots from B' have been shifted by one position to the left compared to B due to the removal of b_t. Consequently:

$$sig(A')[l] = sig(A)[l] \leq sig(B)[l+1] = sig(B')[l].$$

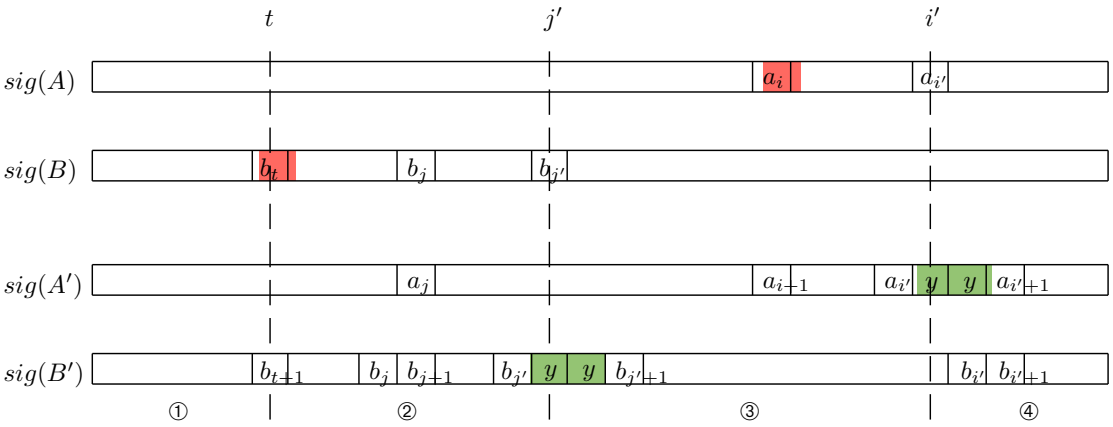

Fig. 4: The various cases of inserting a new interval [x, y] into B and A.

Case 3. $l \in [j', i')$:
Knowing that i' is the position in A where we inserted k new slots with value y, j' is the position in B where we inserted these same slots and $i' \geq j'$, the following is true:

$$sig(A')[l] \leq sig(B)[j'] \leq y = sig(B')[i'] \leq sig(B')[l].$$

Case 4. $l \geq i'$:

a) For $l = i', \ldots, i' + k - 1 : sig(A')[l] = sig(B')[j'] = y$. Since $j' < i' < i' + 1$, then:

$$sig(A')[i'] = y = sig(B')[j'] \leq sig(B')[i'].$$
$$sig(A')[i' + 1] = y = sig(B')[j'] \leq sig(B')[i' + 1].$$

b) For $l \geq i' + k$ the two signatures have equally shifted components compared to the original signatures of A and B, so:

$$sig(A')[l] = sig(A')[i' + k - 1 + (l - i - k + 1)] = sig(A)[i' + (l - i' - k + 1)] \leq$$
$$sig(B)[l - k + 1] = sig(B')[l]$$

In conclusion, $sig(A')[l] \leq sig(B')[l]$ for any l, and relation $A' \preceq B'$ follows. □

Lemma 2. *Given a sequence I of intervals, consider an optimal way of partitioning I into k-ary chains. Let r be a stage where the greedy best-fit algorithm creates a new k-ary chain. Then the optimal way also creates a new k-ary chain.*

Proof: We use the fact that, by Lemma 1, before every step r of the algorithm the multiset Γ_{r-1} of slots created by our greedy algorithm dominates the multiset Ω_{r-1} created by the optimal insertion.

Suppose that at stage r the newly inserted interval I_r causes a new k-ary chain to be created. That means that the left endpoint l_r of I_r is lower than any of the elements of the multiset Γ_{r-1}. By domination, the minimum slot of Ω_{r-1} is at least as high as the minimum slot of Γ_{r-1}. Therefore l_r is also lower than

any slot of Ω_{r-1}, which means that the optimal algorithm also creates a new k-ary chain when inserting I_r. □

Lemma 2 proves that the best-fit algorithm for insertion of new intervals is optimal.

4.2 The interval Hammersley (tree) process

One of the most fruitful avenues for the investigation of the scaling properties of the LIS (Longest increasing subsequence) of a random permutation is made via the study of the (so-called *hydrodynamic*) limit behavior of an interacting particle system known as *Hammersley's process* (Aldous and Diaconis (1995)). This is a stochastic process that, for the purposes of this paper can be defined (in a somewhat simplified form) as follows: random numbers $X_0, X_1, \ldots, X_n, \ldots \in (0,1)$ arrive at integer moments. Each value X_j eliminates ("kills") the smallest $X_i > X_j$ that is still alive at moment j. Intuitively, "live" particles represent the top of the stacks in the *patience sorting algorithm* that computes parameter LIS.

The problem of partitioning a random permutation into a minimal set of k-heapable subsequences is similarly connected to a variant of the above process, introduced in Istrate and Bonchiş (2015) and further studied in Basdevant et al. (2016); Basdevant and Singh (2018), where it was baptized *Hammersley's tree process*. Now particles come with k *lives*, and each particle X_j merely takes one life of the smallest $X_i > X_j$, if any (instead of outright killing it).

The proof of theorem 2 shows that a similar connection holds for sequences of *intervals*. The *Hammersley interval tree process* is defined as follows: "particles" are still numbers in $(0,1)$, that may have up to k lives. However now the sequence $I_0, I_1, \ldots, I_n, \ldots$ is comprised of random *intervals* in $(0,1)$. When interval $I_n = [a_n, b_n]$ arrives, it is a_n that takes a life from the largest live particle $y \leq a_n$. However, it is b_n that is inserted as a new particle, initially with k lives.

Corollary 3. *Live particles in the above Hammersley interval process correspond to slots in our greedy insertion algorithm above. The newly created k-ary chains correspond to local minima (particle insertions that have a value lower than the value of any particle that is alive at that particular moment).*

5 From sequences to sets of intervals

Theorem 2 dealt with *sequences* of intervals. On the other hand a set of intervals does not come with any particular listing order on the constituent intervals. Nevertheless, the problem can be easily reduced to the sequence case by the following:

Theorem 3. *Let $k \geq 1$ and Q be a set of intervals. Then the k-width of Q is equal to the k-width of Gr_Q, the sequence of intervals obtained by listing the intervals in the increasing order of their right endpoints (with earlier starting intervals being preferred in the case of ties).*

Proof: Clearly k-wd(Q)$\leq$ k-wd(Gr_Q), since a partition of Gr_Q into k-ary chains is also a partition of Q.

To prove the opposite direction we need the following:

Lemma 3. *Let S be a multiset of slots. Let $I_1 = [a_1, b_1], I_2 = [a_2, b_2]$ be two intervals such that $b_1 < b_2$, or $b_1 = b_2$ and $a_1 < a_2$. Let S_1 and S_2 be the multisets of slots obtained by inserting the two intervals in the order (I_1, I_2) and (I_2, I_1), respectively. Then S_1 dominates S_2.*

Proof:

The only nontrivial cases are those for which $S_1 \neq S_2$. This condition can only happen when the insertions of I_1, I_2 "interact".

Indeed, let x, y be the values of the largest slots less or equal to a_1, a_2, respectively. If $x \neq y$ and b_1 does not become (after insertion of I_1) the slot occupied by b_2 instead of y (nor does the symmetric situation for the insertion order I_2, I_1 hold) then S_1, S_2 are obtained by adding k copies of b_1, b_2 each, and deleting x, y, so we obtain $S_1 = S_2$.

If instead $x = y$ then the two slots to be deleted (for both insertion orders) are x and the largest slot smaller or equal than x. We obtain again $S_1 = S_2$.

The only remaining case is when the slot removed after insertion of I_2 is b_1. In this case S cannot contain any element in the range (b_1, a_2). Insertion (I_1, I_2) removes x, adds $k-1$ copies of b_1 and k copies of b_2. On the other hand, insertion (I_2, I_1) may remove some element x'. It is certainly $x \leq x' \leq b_1$. It then adds k copies of b_2. Then it removes some $x'' \leq x$ and adds k copies of b_1. Thus both sets S_1, S_2 can be described as adding k copies of b_1 and k copies of b_2 to S, and then deleting one or two elements, two of them

- x and b_1 in the case of order (I_1, I_2)
- x' and x'' in the case of order (I_2, I_1).

in the case when some element of S is strictly less than a_1, and one element

- b_1 in the case of order (I_1, I_2)
- x'' in the case of order (I_2, I_1)

otherwise. In the second case the result follows by taking into account the fact that $x'' \leq b_1$ and the following lemma:

Lemma 4. *Let S be a multiset of slots. Let $s_2, s_1 \in S$ with $s_2 \leq s_1$, and let S_1, S_2 be the sets obtained by deleting from S elements s_1 (or s_2, respectively). Then S_1 dominates S_2.*

Proof: It follows easily from the simple fact that for every $r \geq 1$ and multiset W the function that maps an integer $x \in W$ to the r'th smallest element of $W \setminus x$ is non-increasing (more precisely a function that jumps from the $r+1$'th down to the r'th smallest element of W) $\square$ The first case is only slightly more involved. Since $x' \leq x$, by applying Lemma 4 twice we infer:

$$\begin{aligned} S_1 = \quad & S \setminus \{x, b_1\} = (S \setminus \{b_1\}) \setminus \{x\} \preceq (S \setminus \{b_1\}) \setminus \{x'\} = (S \setminus \{x'\}) \setminus \{b_1\} \preceq \\ \preceq \quad & (S \setminus \{x'\}) \setminus \{x''\} = S_2. \end{aligned}$$

which is what we wanted to prove. $\square$

From Lemma 3 we infer the following result

Lemma 5. *Let $1 \leq r \leq n$ and let X, Y be two permutations of intervals $I_1, I_2, \ldots, I_n$,*

$$\begin{aligned} X &= (I_1, \ldots, I_{r-1}, I_r, I_{r+1}, \ldots I_n), \\ Y &= (I_1, \ldots, I_{r-1}, I_{r+1}, I_r, \ldots I_n). \end{aligned}$$

Input: A set of intervals I.
Output: A partition H of I into k-ary chains.
Sort the intervals w.r.t. $\sqsubseteq$: $I = (I_1, \ldots, I_n)$.
For $i := 1$ to n do:

If $I_i = [a_i, b_i]$ can be inserted into some empty slot

then insert I_i in the highest-valued compatible slot.

else create a new k-chain rooted at I_i

Fig. 5: The greedy algorithm for sets of intervals.

respectively (i.e. X, Y differ by a transposition). If $I_r \leq I_{r+1}$ (recall, this means that the right endpoint of I_r is less or equal than the left endpoint of I_{r+1}) then multisets of slots S_X, S_Y obtained by inserting intervals according to the listing specified by X and Y, respectively, satisfy

$$S_X \preceq S_Y.$$

Proof: Without loss of generality one may assume that $r = n - 1$ (as the result for a general n follows from this special case by repeatedly applying Lemma 1). Let S be the multiset of slots obtained by inserting (in this order) intervals $I_1, \ldots, I_{r-1}$. Applying Lemma 3 to intervals I_r, I_{r+1} we complete the proof of Lemma 5. □

Now the opposite direction in the proof of Theorem 3 follows: the multiset $S_{Gr(Q)}$ of labels obtained by inserting the intervals according to sequence Gr_Q dominates any multiset of labels arising from a different permutation, since one can "bubble down" smaller intervals (as in bubble sort), until we obtain Gr_Q. As we do so, at each step, the new multiset of labels dominates the old one. Hence $S_{Gr(Q)}$ dominates all multisets arising from permutations of $I_1, \ldots, I_n$, so sequence Gr_Q minimizes the parameter k-wd among all permutations of Q. □

Corollary 4. *The greedy algorithm in Figure 5 computes the k-width of an arbitrary set of intervals.*

Corollary 5. *Modify the Hammersley interval process to work on sets of intervals by considering them in non-decreasing order according to relation $\sqsubseteq$. Then live particles in the modified process correspond to slots obtained using our greedy insertion algorithm for sets of intervals. New k-ary chains correspond to local minima (such particle insertions that have a value lower than the value of any particle that is alive at that particular moment).*

6 Extension to sequences of elements from a trapezoid partial order

The theorem in the previous section is strongly reminiscent of the fact that for interval partial orders a greedy best-fit algorithm computes the chromatic number (Olariu (1991)). This result has an extension to

an even more general class of graphs, that of *trapezoid graphs* (Dagan et al. (1988)). Trapezoid graphs are an extension of both interval and permutation graphs that unify many natural algorithms (for problems such as maximum independent set, coloring) for the two classes of graphs.

As noted in Felsner et al. (1997), trapezoid graphs can be equivalently defined as the cocomparability graphs of two-dimensional boxes, with sides parallel to the coordinate axes:

Definition 10. *A* box *is the set of points in* $\mathbf{R}^2$ *defined as*

$$B = \{(x_1, x_2) \in \boldsymbol{R}^2 \mid l_i^B \leq x_i \leq r_i^B, i = 1, 2\}$$

for some numbers $l_i^B \leq r_i^B \in \boldsymbol{R}$*, where* $l_B = (l_1^B, l_2^B)$ *is* the lower corner *of the box* B *and* $r_B = (r_1^B, r_2^B)$ *is* the upper corner.

The interval $I(B)$ associated to box B *is the projection onto the* y *axis of box* B*. Clearly* $I(B) = [l_2^B, r_2^B]$.

The dominance partial order on intervals naturally extends to boxes:

Definition 11. *The* dominance partial order among boxes *is defined as follows: box* B_1 *dominates box* B_2 *if point* r_{B_1} *dominates point* l_{B_2}*, i.e.*

$$r_i^{B_1} \leq l_i^{B_2} \text{ for } i = 1, 2.$$

A trapezoid partial order is a poset P *induced by the dominance partial order on a finite set* V *of boxes.* l *and* u *refer to the lower and upper corner coordinate functions defining the boxes. That is, for every* $v \in V$*,* $l(v)$ *is the lower corner of box* v *and* $u(v)$ *is its upper corner.*

The box representation allowed the authors of Felsner et al. (1997) to give a "sweep-line algorithm" for coloring trapezoid graphs that improved the coloring algorithm in Dagan et al. (1988).

As noted in Section 2.1, partition into k-heapable sequences is a natural generalization of coloring permutation graphs. In the sequel we give an algorithm that generalizes the sweep-line coloring algorithm for trapezoid graphs from Dagan et al. (1988) to the partition into k-ary chains of a particular class of sequences of boxes.

Theorem 4. *Let* $\mathcal{B} = (B_1, B_2, \ldots, B_n)$ *be a sequence of two-dimensional axis-parallel boxes, totally ordered by the* x*-coordinates of their right endpoints. Then the greedy sweep-line algorithm in Figure 6 computes an optimal partition of sequence* $\mathcal{B}$ *into* k*-ary chains.*

Proof: The proof of Theorem 4 leverages and extends the method used for intervals in the proof of Theorem 2. The fact that sequence $\mathcal{B}$ is sorted in increasing order of the x-coordonate of the endpoints allows us to see the problem as one on intervals: instead of dealing with boxes $B_1, B_2, \ldots, B_n$ we will instead deal with the associated intervals $I(B_1), I(B_2), \ldots, I(B_n)$.

We will assume that the x and the y coordinates of all boxes in $\mathcal{B}$ are all distinct. As usual with such algorithms, the truth of our statement does not rely on this assumption: if the statement is not true then points can be slightly perturbed to make the assumption true. The algorithm we give then extends to the degenerate cases as well.

The sweep line maintains a multiset S of *slots*, that correspond to right endpoints of the intervals associated with the boxes processed so far. The difference with respect to the case of Theorem 2 is that the slots are now of one of two types:

Input: A sequence of boxes $\mathcal{B} = (B_1, B_2, \ldots, B_n)$.
Output: A partition H of $\mathcal{B}$ into k-ary chains.

let $\mathcal{P} = \{(p_1, p_2) | (p_1, p_2) = l_{B_i} \text{or} (p_1, p_2) = u_{B_i}, i \in 1...n\}$

initialize $S = \{d\}$, where d is a real number smaller than all

the y coordinates of points $p \in \mathcal{P}$, marked *available*

foreach $p \in \mathcal{P}$ sorted increasingly by the second coordinate **do**:

$q \leftarrow$ first available slot below p_2 in S

if $p = l(v)$ for some $v \in \mathcal{B}$ **then**

if $q = u(w)_2$ for some $w \in \mathcal{B}$ **then**

insert v in the k-ary chain of w as a child of this node
remove q from S
add k copies of p_2 to S, marking them unavailable

else // $(q == d)$

start a new k-ary chain rooted at v
add k copies of p_2 to S, marking them unavailable

if $p = u(v)$ for some $v \in \mathcal{B}$ **then**

mark all slots with value p_2 in S as available

return the set of k-ary chains constructed by the algorithm.

Fig. 6: The greedy sweep-line algorithm for x-sorted trapezoid sequences.

- *unavailable:* a slot of this type is present in multiset S but cannot be used to process intervals.
- *available:* a slot of this type can be fully employed when processing a new interval.

The action of a sweep line comprises two types of actions:

- *insertion:* This happens when the sweep line reaches the left corner of some box B_j. We process the interval $I(B_j)$ similarly to the process in Theorem 2. Namely, we remove one lifeline from the largest **available slot** with value at most $l_2^{B_j}$, and insert into S k slots with value $r_2^{B_j}$. These slots are marked **unavailable.**
- *state change:* This happens when the sweep line reaches the right corner of some box B_j. All the slots with value $r_2^{B_j}$ change marking, from unavailable to **available.**

The intuitive explanation for this modification is clear: a box D can become the parent of another box E only when the right corner of D is to the left (on the x axis) of the left corner of E. Thus, if the sweep line has not reached the right endpoint of a box D then this box is not eligible to become the parent of any currently processed box.

Let σ_t be the multiset of all slots created by the sweep-line algorithm up to a given moment t. Let OPT_t be the multiset of slots corresponding to an *optimal* partition into k-ary chains. Let $\sigma_t^{'}$ be the corresponding (multi)set of all slots in σ_t marked *available* at moment t. Let $OPT_t^{'}$ be the set of slots in OPT_t marked available at moment t.

The basis for the proof of Theorem 4 is the following adaptation of Lemma 1 to the setting of slots with availability statuses:

Lemma 6. *For every $t \geq 0$, $sig(\sigma'_t) \preceq sig(OPT'_t)$.*

Proof: By induction on t. The statement is clear for $t = 0$, since both σ_0 and OPT_0 are empty. Assume the claim is true for all $t' < t$. Let $sig(\sigma'_{t-1}) = [a_1, a_2, \ldots, a_{|sig(\sigma'_{t-1})|}]$ and $sig(OPT'_{t-1}) = [b_1, b_2, \ldots, b_{|sig(OPT'_{t-1})|}]$. Also, by convention, define $a_0 = b_0 = -\infty$ and $a_{|sig(\sigma'_{t-1})|+1} = b_{|sig(OPT'_{t-1})|+1} = +\infty$. Finally, let $[l_s^2, r_s^2]$ is the interval to be processed at stage t.

Proving that $\sigma'_{t-1} \preceq OPT'_{t-1}$ entails proving that $|sig(\sigma'_t)| \leq |sig(OPT'_t)|$ and for all indices $1 \leq l \leq |sig(\sigma'_t)|$:

$$sig(\sigma'_t)[l] \leq sig(OPT'_t)[l]. \tag{3}$$

- **Case 1: t is an insertion step:**

  σ'_t is obtained from σ'_{t-1} by removing the largest available slot less or equal to l_s^2. OPT'_t is obtained from OPT'_{t-1} by perhaps removing some slot with value at most l_s^2.

  The inequality $|\sigma'_t| \leq |OPT'_{t-1}|$ follows easilty from the corresponding inequality at stage $t-1$. Indeed, σ'_{t-1} may lose an element (in which case OPT'_{t-1} may also lose at most one element) or stay the same (in which case OPT'_{t-1} also stays the same (because there is no available slot to lose a lifeline.)

  As for inequality (3): if OPT'_{t-1} does not lose an element the inequality follows easily from the induction hypothesis for stage $t-1$ and the fact that elements of σ'_{t-1} stay in place or shift to the left.

  If both σ'_{t-1} and OPT'_{t-1} lose one element to yield σ'_t, OPT'_t, there are four types of positions l:

  - Those below both deleted positions. They are unchanged as we move from σ'_{t-1} and OPT'_{t-1} to σ'_t, OPT'_t. That is:

    $$\sigma'_t[l] = \sigma'_{t-1}[l] \text{ and } OPT'_t[l] = OPT'_{t-1}[l].$$

    Hence inequality (3) is true by the induction hypothesis.

  - Those above both deleted positions. They get shifted to the left by one in both σ'_t and OPT'_t. That is:

    $$\sigma'_t[l] = \sigma'_{t-1}[l+1] \text{ and } OPT'_t[l] = OPT'_{t-1}[l+1].$$

    Hence again inequality (3) is true by the induction hypothesis.

– Those below one but above the other deleted position. By the fact that the deleted slot in OPT'_{t-1} is less or equal than the one in σ'_{t-1} (since this is the largest available slot less or equal to l_s^2), it follows that

$$\sigma'_t[l] = \sigma'_{t-1}[l] \text{ and } OPT'_t[l] = OPT'_{t-1}[l+1].$$

Hence

$$\sigma'_t[l] = \sigma'_{t-1}[l] \leq OPT'_{t-1}[l] \leq OPT'_{t-1}[l+1] = OPT'_t[l].$$

so relation (3) holds in all cases.

- **Case 2: t is a change step:** The effect of such a step is that, both in σ_{t-1} and OPT_{t-1} the k slots with value r_s^2 (that were previously inserted, but marked unavailable) become available.

  Since $|\sigma'_t| = |\sigma'_{t-1}| + k$ and, similarly, $|OPT'_t| = |OPT'_{t-1}| + k$, statement $|\sigma'_t| \leq |OPT'_t|$ follows from the analogous statement for $t-1$.

  Positions l before the insertion points of the k copies of r_s^2 are not modified in σ'_t, OPT'_t so equation (3) follows for such l's from the corresponding inequalities for stage $t-1$. Similarly, positions larger than both insertion points get shifted by exactly k, so inequality (3) also follows for such l's by the corresponding inequality for stage $t-1$.

  These two cases cover all positions l for which both values are different from the newly inserted values equal to r_s^2. If *both* positions are equal to r_s^2 the inequality also follows. The only remaining cases are those l for which one of $\sigma'_t[l], OPT'_t[l]$ is equal to the newly inserted value r_s^2 but the other is not. On inspection, though, inequality (3) is true in these cases as well: because of dominance, the insertion point into OPT'_{t-1} is to the left (or equal) to the insertion point into σ'_{t-1}. So if $\sigma'_t[l] \neq r_s^2$ but $OPT'_t[l] = r_s^2$ then $\sigma'_t[l] < r_s^2 = OPT'_t[l]$. The reasoning in the case $\sigma'_t[l] = r_s^2$ but $OPT'_t[l] \neq r_s^2$ is analogous, the conclusion being that $\sigma'_t[l] = r_s^2 < OPT'_t[l]$.

□

Using Lemma 6 we infer that at every step t when the greedy sweep-line algorithm creates a new k-ary chain (because there is no available slot to lose a lifeline), so does the optimal algorithm (for the very same reason). Hence the greedy sweep-line algorithm is optimal. □

7 Maximal heapable subsets of interval orders

Finally, note that we are unable to solve the open problem from Byers et al. (2011) on the complexity of computing a maximum heapable subsequence, i.e. a maximum treelike 2-independent set in permutation posets (see the definition and discussion in Section 2.1). Instead, we settle this problem for a different subclass of partial orders, the class of interval orders:

Theorem 5. *Given a set of intervals I, the best-fit algorithm in Figure 7 computes a largest k-heapable subset of intervals of the set I.*

Proof:

Note first that, whereas the theorem deals with *subsets* of the set of intervals, the algorithm in Figure 7 considers the intervals in the set in a specific order (by sorting them with respect to $\sqsubseteq$). This will turn not

Input: A set of intervals I.
Output: A k-ary chain $J \subseteq I$ of maximum cardinality.

Sort the intervals w.r.t. $\sqsubseteq$: $SI = (I_1, \ldots, I_n)$.
Let $J = \emptyset$.
For $i := 1$ to n do:

If $I_i = [l_i, r_i]$ can be inserted into some empty slot

then

$J = J \cup \{i\}$.

insert I_i in the highest-valued compatible slot.

Fig. 7: The greedy best-fit algorithm for sets of intervals.

to be a problem, because we prove by induction the following result, which implies the statement of the theorem:

Lemma 7. *For every $1 \leq r \leq n$ there exists a largest k-heapable subset Γ_r of $\{I_1, I_2 \ldots, I_n\}$ and a k-heap T_r for Γ_r such that the tree T_G built by the greedy algorithm of Figure 7 agrees with T_r with respect to the presence or absence of intervals $I_1, I_2 \ldots, I_r$.*

Proof: By induction.

The result is simple for $r = 1$: if I_1 is part of an optimal k-heapable subset then there is nothing to prove, since I_1 can only participate in a k-heapable subset as the root of its corresponding tree. This happens because $I_1 \sqsubseteq I_s$ for all $s \geq 1$. If, on the other hand, I_1 is not part of an optimal subset, then consider such a solution Γ. Create a new solution Γ_1 with the same tree shape by replacing the root interval I of Γ by I_1 (thus obtaining tree T_1). This is legal, since I_1 ends earlier than I.

Assume we have proved the result for $1 \leq i \leq r-1$. Consider now the optimal solution Γ_{r-1} (with k-ary tree T_{r-1}) agreeing with the greedy solution T_G on the presence of the first $r-1$ intervals.

- If I_r cannot be inserted into T_G (that is, in the portion of T_G constructed by the greedy-best fit algorithm after considering intervals $I_1, I_2, \ldots, I_{r-1}$) it means that no slot is available for I_r. By domination (Lemma 1) this must be true for both the greedy and the optimal solution Γ_{r-1}. Therefore these two solutions do not contain I_r, hence they agree on the presence or absence of $I_1, I_2, \ldots, I_r$.

- Suppose now that I_r can be inserted in a slot (thus is present in the greedy solution) but is **not** present in the optimal solution Γ_r. Let J be the interval that has the same parent in Γ_r as I_r has in the greedy solution (Figure 8). J must exist, otherwise one could simply extend Γ_{r-1} by inserting I_r at that position.

  By the order we considered, the intervals I_r ends no later than J does. So by inserting I_r instead of J we obtain an optimal solution $\Gamma_r = \Gamma_{r-1} \setminus \{J\} \cup \{I_r\}$ that satisfies the induction property.

□

The conclusion of the induction argument is that there exists an optimal tree containing *exactly the same* intervals as those selected by T_G. Thus the greedy best-fit algorithm produces an optimal solution. □

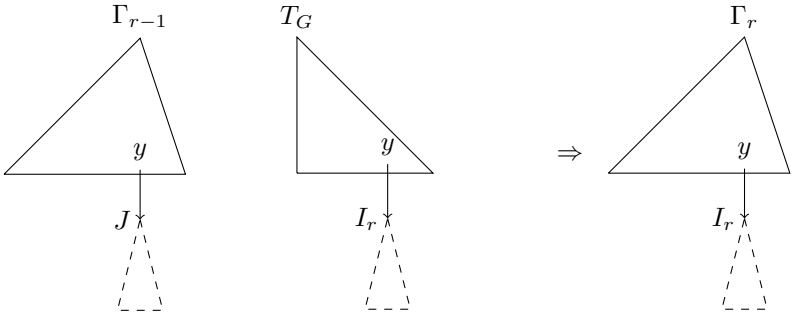

Fig. 8: Creating optimal solution $\Gamma_r = \Gamma_{r-1} \setminus \{J\} \cup \{I_r\}$.

8 Open questions and future work

Our Theorem 1 is very similar to (the proof of) Dilworth's theorem. It is not *yet* a proper generalization of this result to the case $k \geq 1$ because of the lack of a suitable extension of the notion of *antichain*:

Open problem 1. *Is there a suitable definition of the concept of k-antichain, that coincides with this concept for $k = 1$ and leads (via our theorem 1) to an extension of Dilworth's theorem?*

On the other hand, results in Section 4 naturally raise the following:

Open problem 2. *For which partial orders Q can one compute the parameter k-wd(Q) (and an associate optimal k-ary chain decomposition) via a greedy algorithm?*

Several open problems concern the limit behavior of the expected value of the k-width of a set of random intervals, for $k \geq 1$. As discussed in Section 2, for $k = 1$ the scaling behavior of this parameter is known (Justicz et al. (1990)). However, in the case of random permutations, the most illuminating description of this scaling behavior is by analyzing the hydrodynamic limit of the Hammersley process (Aldous and Diaconis (1995); Groeneboom (2002)). As shown in Section 5, the difference between sequences and sets of intervals is not substantial.

We ask, therefore, whether the success in analyzing this process for random permutations can be replicated in the case of sequences/sets of random intervals:

Open problem 3. *Analyze the hydrodynamic limit of the Hammersley process for sequences/sets of random intervals.*

When $k \geq 2$ even the scaling behavior is not known, for both sequences and sets of random intervals. The connection with the interval Hammersley process given by Corollaries 3 and 5 provides a convenient, lean way to simulate the dynamics, leading to experimental observations on the scaling constants. A C++ program used to perform these experiments is publicly available at Istrate (2017). Based on these experiments we would like to raise the following:

Conjecture 1. *For every $k \geq 2$ there exists a positive constant $c_k > 0$ such that, if R_n is a sequence of n random intervals then*

$$\lim_{n\to\infty} \frac{E[k\text{-}wd(R_n)]}{n} = c_k \tag{4}$$

Moreover $c_k = \frac{1}{k+1}$.

According to this conjecture, just as in the case of random permutations, the scaling behavior of the k-width changes when going from $k = 1$ to $k = 2$. Note, though, that the direction of change is different ($\Theta(\sqrt{n})$ to $\Theta(\log n)$ for integer sequences, $\Theta(\sqrt{n})$ to $\Theta(n)$ for sequences of intervals). On the other hand, somewhat surprisingly, the scaling behavior of sets of random intervals seems to be similar to that for sequences:

Conjecture 2. *For every $k \geq 2$ there exists a positive constant $d_k > 0$ such that, if W_n is a* **set** *of n random intervals then*

$$\lim_{n\to\infty} \frac{E[k\text{-}wd(W_n)]}{n} = d_k \tag{5}$$

Experiments suggest that $d_2 = c_2 = \frac{1}{3}$, and similarly for $k = 3, 4$ $d_k = c_k = \frac{1}{k+1}$. Therefore we conjecture that

$$d_k = c_k = \frac{1}{k+1} \text{ for all } k \geq 2. \tag{6}$$

References

D. Aldous and P. Diaconis. Hammersley's interacting particle process and longest increasing subsequences. *Probability theory and related fields*, 103(2):199–213, 1995.

A.-L. Basdevant and A. Singh. Almost-sure asymptotic for the number of heaps inside a random sequence. *Electronic Communications in Probability*, 23(17), 2018.

A.-L. Basdevant, L. Gerin, J.-B. Gouéré, and A. Singh. From Hammersley's lines to Hammersley's trees. *Probability Theory and Related Fields*, pages 1–51, 2016.

G. E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, Nov. 1990.

C. Bonchiş, G. Istrate, and V. Rochian. The language (and series) of Hammersley-type processes. In *Proceedings of the Eighth Conference on Machines Computation and Universality (MCU'18)*, pages 69–87, 2018.

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.

J. Byers, B. Heeringa, M. Mitzenmacher, and G. Zervas. Heapable sequences and subseqeuences. In *Proceedings of the Eighth Workshop on Analytic Algorithmics and Combinatorics (ANALCO'2011)*, pages 33–44. SIAM Press, 2011.

K. Chandrasekaran, E. Grigorescu, G. Istrate, S. Kulkarni, Y.-S. Lin, and M. Zhu. The maximum binary tree problem. *arXiv preprint arXiv:1909.07915*, 2019.

H. Crane and S. DeSalvo. Pattern Avoidance for Random Permutations. *Discrete Mathematics & Theoretical Computer Science*, Vol. 19 no. 2, Permutation Patterns 2016, Dec. 2018. doi: 10.23638/DMTCS-19-2-13. URL `https://dmtcs.episciences.org/5011`.

I. Dagan, M. C. Golumbic, and R. Y. Pinter. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21(1):35–46, 1988.

C. Defant. Proofs of Conjectures about Pattern-Avoiding Linear Extensions. *Discrete Mathematics & Theoretical Computer Science*, vol. 21 no. 4, Oct. 2019. doi: 10.23638/DMTCS-21-4-16. URL `https://dmtcs.episciences.org/5796`.

R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, pages 161–166, 1950.

K. Durant and S. Wagner. On the centroid of increasing trees. *Discrete Mathematics & Theoretical Computer Science*, vol. 21 no. 4, Aug. 2019. doi: 10.23638/DMTCS-21-4-8. URL `https://dmtcs.episciences.org/5691`.

J. Egerváry. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38(1931):16–28, 1931.

S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics*, 74(1):13–32, 1997.

S. Felsner, V. Raghavan, and J. Spinrad. Recognition algorithms for orders of small width and graphs of small Dilworth number. *Order*, 20(4):351–364, 2003.

D. R. Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. *Proceedings of the American Mathematical Society*, 7(4):701–702, 1956.

C. Greene and D. J. Kleitman. The structure of Sperner k-families. *Journal of Combinatorial Theory, Series A*, 20(1):41–68, 1976.

P. Groeneboom. Hydrodynamical methods for analyzing longest increasing subsequences. *Journal of Computational and Applied Mathematics*, 142(1):83–105, 2002.

A. Gyárfás and J. Lehel. Covering and coloring problems for relatives of intervals. *Discrete Mathematics*, 55(2):167–180, 1985.

G. Istrate. C++ program *interval* for experiments on the heapability of intervals. `https://bitbucket.org/gabriel-istrate/heapable-intervals`, 2017. Last accessed: May 12, 2020.

G. Istrate and C. Bonchiş. Partition into heapable sequences, heap tableaux and a multiset extension of Hammersley's process. In *Proceedings of the 26th Annual Symposium on Combinatorial Pattern Matching (CPM'15), Ischia, Italy*, volume 9133 of *Lecture Notes in Computer Science*, pages 261–271. Springer, 2015.

G. Istrate and C. Bonchiş. Heapability, interactive particle systems, partial orders: Results and open problems. In *Proceedings of the 18th International Conference on Descriptional Complexity of Formal Systems (DCFS'2016), Bucharest, Romania*, volume 9777 of *Lecture Notes in Computer Science*, pages 18–28. Springer, 2016.

J. Justicz, E. R. Scheinerman, and P. M. Winkler. Random intervals. *Amer. Math. Monthly*, 97(10): 881–889, 1990.

H. A. Kierstead, G. F. McNulty, and W. T. Trotter. A theory of recursive dimension for ordered sets. *Order*, 1(1):67–82, 1984.

D. Konig. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

C. L. Mallows. Patience sorting. *SIAM Review*, 5(4):375–376, 1963.

S. Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37(1): 21–25, 1991.

J. Porfilio. A combinatorial characterization of heapability. Master's thesis, Williams College, May 2015. Available from `https://unbound.williams.edu/theses/islandora/object/studentttheses%3A907`. Last accessed: May 12, 2020.

D. Romik. *The surprising mathematics of longest increasing subsequences*. Cambridge University Press, 2015.

W. T. Trotter. Partially ordered sets. *Handbook of combinatorics*, 1:433–480, 1995.