Searching in Random Partially Ordered Sets

R. Carmo [a,b,1,*] J. Donadelli [a,2] Y. Kohayakawa [b,3] E. Laber [c,4]

[a] *Departamento de Informática — Universidade Federal do Paraná*
Centro Politécnico da UFPR, 81531–990, Curitiba PR, Brazil

[b] *Instituto de Matemática e Estatística — Universidade de São Paulo*
Rua do Matão 1010, 05508–090 São Paulo SP, Brazil

[c] *Departamento de Informática — Pontifícia Univ. Católica do Rio de Janeiro*
R. Marquês de São Vicente 225, Rio de Janeiro RJ, Brazil

Abstract

We consider the problem of searching for a given element in a partially ordered set. More precisely, we address the problem of computing efficiently near-optimal search strategies for typical partial orders under two classical models for random partial orders, the *random graph model* and the *uniform model*.

We shall show that the problem of determining an optimal strategy is $\mathcal{NP}$-hard, but there are simple, fast algorithms able to produce nearly-optimal search strategies for typical partial orders under the two models of random partial orders that we consider. We present a $(1+o(1))$-approximation algorithm for typical partial orders under the *random graph* model (constant p) and present a 6.34-approximation algorithm for typical partial orders under the *uniform* model. Both algorithms run in polynomial time.

Key words: Searching, search trees, random partial orders

* Corresponding author
Email addresses: renato@ime.usp.br (R. Carmo), jair@inf.ufpr.br (J. Donadelli), yoshi@ime.usp.br (Y. Kohayakawa), laber@inf.puc-rio.br (E. Laber).

[1] Partially supported by PICDT/CAPES.
[2] Supported by a CNPq doctorate Studentship (Proc. 141633/1998–0).
[3] Partially supported by MCT/CNPq through ProNEx Programme (Proc. CNPq 664107/1997–4) and by CNPq (Proc. 300334/93–1 and 468516/2000–0).
[4] Partially supported by FAPERJ through "Jovem Cientista" Program (E-26/150.362/2002) and CNPq.

1 Introduction

A fundamental problem in data structures is the problem of representing a dynamic set S in such a way that we may perform search operations efficiently.

Perhaps the most common assumption about the set S is that its elements belong to some *totally ordered set* U. In this paper, we are interested in examining a certain variant of this problem, where the 'universe' U from which our elements are drawn is a *partially ordered set*. The reader is referred to [4] for the motivation for considering this problem (see also Section 2 below).

Our first result will be negative: we shall prove that this problem is $\mathcal{NP}$-hard in general, answering a question raised in [4]. We shall then address the problem of efficiently computing near-optimal search strategies for typical partial orders under two classical models for random partial orders: the *random graph model* and the *uniform model*. For these 'relaxed' versions of the problem, one is able to prove fairly strong results quite easily.

The problem we consider generalizes the problem of searching through a totally ordered domain, for which the well known binary search strategy is the optimal solution. The best way to formulate our problem is perhaps by making use of a 2-player game, which we now describe.

A 2-player game. Let a partial order $\mathcal{U} = (U, \prec)$ and a set $S \subseteq U$ be given. The two players of our game $\Gamma(\mathcal{U}, S)$ are the hider and the seeker. The hider initially chooses an element $u \in U$ and the seeker has to *search* for this element until he *finds* it.

A move of the seeker is simply to pick an element s from S, which is interpreted as the question "*is s the chosen element u?*". On being presented this question, the hider replies either hit, meaning "*Yes, it is*", smaller, meaning "*No, but $u \prec s$*", or no, meaning "*No, and $u \not\preceq s$*". A sequence of moves, or *queries*, made by the seeker along the game will be called a *search*, and an algorithm that decides the next query based on the past will be called a search strategy.

The game ends when the seeker *finds* the chosen element, that is, the seeker receives a hit as the answer, or else he is in position to declare that $u \notin S$ with certainty. The goal of the seeker is to finish the game as soon as possible, and the goal of the hider is to delay the end of the game for as long as possible, by choosing "his best u". (In fact, since u does not have to be disclosed until the very end, the hider does not have to make up his mind about which u to pick at the beginning; he may answer the queries as the game evolves, just making sure that his answers are consistent with *some* choice of u.)

We define an *optimal search strategy* for $\Gamma(\mathcal{U}, S)$ as a search strategy in which the longest search is as short as possible.

Let us recall that our game $\Gamma(\mathcal{U}, S)$ is defined based on an order $\mathcal{U} = (U, \prec)$ and a set $S \subseteq U$. The reason to have both U and S, instead of having just S (with the induced order), goes back to the motivation of our game: with this slightly more cumbersome definition, we are able to model the case of *unsuccessful* searches in data structures. However, the reader will see below that, in fact, the larger set U will not really play any rôle once the problem is suitably formalized.

The computational problem. Having introduced the game $\Gamma(\mathcal{U}, S)$, we may now state a related computational problem. The problem of *searching through a partially ordered set* (SPOS) is the problem of devising an optimal search strategy for $\Gamma(\mathcal{U}, S)$. More precisely, given a directed graph G_P representing the Hasse diagram of the order P induced by $\mathcal{U}$ on S, we wish to compute an optimal search strategy for $\Gamma(\mathcal{U}, S)$. Problem SPOS may be summarized as follows:

(1) an instance is a directed graph G_P, representing the Hasse diagram of a partial order $P = (S, \prec)$, and
(2) a solution is an optimal search strategy for $\Gamma(\mathcal{U}, S)$.

The results. We shall first prove that SPOS is $\mathcal{NP}$-hard, in answer to the main question raised in [4]. Then we shall show that certain simple, fast algorithms are able to produce nearly-optimal search strategies for typical instances under the *random graph* and the *uniform* models of random partial orders (see Sections 4 and 5 for definitions). The reader is referred to [7] for an excellent survey on these models.

We write n for the number of elements in the order $P = (S, \prec)$ and, as usual, we use the expression "*almost surely*" to mean "*with probability tending to* 1 *as* $n \to \infty$". We also use the common terms "*almost every*" and "*almost all*".

We shall consider the *random graph model* with constant p. We present a polynomial time algorithm that produces a search strategy that makes at most

$$\log_2 n + O\left((\log n)^{1/2} \log\log n\right)$$

queries in the worst case for almost every n-element order P in this model.

In the *uniform model*, the situation is somewhat different: with the help of the fundamental result of [12], it is easy to see that almost all n-element partial orders are such that *any* search strategy makes at least about $n/4$ queries in

the worst case. We therefore consider a slightly more 'generous' hider, who replies whether or not $s = u$, and if this is not the case then tells the seeker whether $u \prec s$, $s \prec u$, or that u is not comparable with s. With this more generous hider, almost all partial orders P admit search strategies that require at most

$$(6.33\ldots + o(1)) \log_3 n$$

queries. We shall present a polynomial time algorithm to produce such a strategy.

Since we need to make at least $\log_2 n$ queries for *any* n-element partial order ($\log_3 n$ in the 'generous' hider version), our results tell us that one may efficiently devise near-optimal search strategies for almost all partial orders in the first model, and one may efficiently devise search strategies for almost all partial orders in the second model that are worse than the optimal by a constant factor only.

We now introduce the notation and the formal definitions that we shall use.

1.1 Problem Statement and Notation

A *partial order* is a pair $P = (S, \prec)$, where S is a set and $\prec$ is a binary relation on S that is irreflexive, anti-symmetric, and transitive. If x, $y \in S$ then $x \prec y$ stands for $(x, y) \in \prec$. If X is a subset of S, we let $P(X) = (X, \prec \cap X^2)$ and $P - X = P(S - X)$. For the remainder of this section, $P = (S, \prec)$ will denote an order and X will denote a subset of S.

An *ideal* of P is a set $I \subseteq S$ such that, for all $i \in I$, if $s \prec i$ then $s \in I$. We write $I_P(X)$ for the minimal ideal I of P such that $X \subseteq I$ and write $I_P^-(X)$ for $I_P(X) - X$. Dualizing, we obtain the notion of filters. Let $P^* = (S, \prec')$ be the *dual order* to P, that is, the order with $x \prec' y$ if and only if $y \prec x$. An ideal in P^* is called a *filter* of P. Moreover, we let $F_P(X) = I_{P^*}(X)$ and $F_P^-(X) = I_{P^*}^-(X)$. When the order P is clear from the context, we omit the subscript P.

A *query about* $u \in U$ *to* $s \in S$ has three possible outcomes, namely, hit, smaller and no, meaning, respectively, $u = s$, $u \prec s$ and $u \not\preceq s$. A *search for* $u \in U$ *through* S is a sequence of queries terminating with hit, or else with a situation in which one may deduce that $u \notin S$.

Consider a search for u through S as above. At any given point of the search, we have a set $X \subseteq S$ of 'candidates' and we must choose some $s \in S$ to query about u. Suppose an $s \neq u$ is chosen. Then, once the query is answered, the set of 'candidates' is reduced to $X \cap I^-(s)$ in the case the answer is smaller, and is reduced to $X - I(s)$ in the case the answer is no.

Our goal is to devise a search strategy in which the longest search poses the smallest number of queries. Such a strategy may be conveniently thought of as a binary search tree whose nodes are labelled with elements of S and whose edges are labelled with smaller and no, and furthermore has the smallest possible height.

In what follows we state some definitions in order to make the above more precise. As usual, a *binary tree* T is either the empty tree Λ, with no *nodes*, or else $T = (r, T_L, T_R)$, where r is the *root node* of T and T_L and T_R are its *left* and *right subtrees*, which are trees with fewer nodes than T. The *height* $h(T)$ of a binary tree T is 0 if $T = \Lambda$, and is $1 + \max\{h(T_L), h(T_R)\}$ if $T = (r, T_L, T_R)$. The *rightmost path* of a binary tree $T = (r, T_L, T_R)$ is the path starting at r, followed by the rightmost path of T_R. The rightmost path of Λ is the empty path.

A *binary search tree T for X with respect to P* is the empty binary tree Λ if $X = \emptyset$, and if $X \neq \emptyset$, then $T = (s, T_Y, T_N)$, where s is some element of S, and T_Y and T_N are binary search trees for $X \cap I^-(s)$ and $X - I(s)$, respectively. (The set X should be thought of as the set of 'candidate elements' in S, as the game between hider and seeker evolves.) See Figure 2 for an example of a simple binary search tree.

An *optimal tree for X with respect to P* is a binary search tree for X with respect to P of minimal height. If we write $h^*_P(X)$ for the height of such a tree, it is immediate from the above definitions that, for some $s \in S$, we have

$$h^*_P(X) = 1 + \max\left\{h^*_P(X \cap I^-(s)), h^*_P(X - I(s))\right\}. \tag{1}$$

A *binary search tree for $P = (S, \prec)$* is a tree for S with respect to P. An *optimal tree* for P is a tree of minimal height for P. Writing $h^*(P)$ for this minimum, as in (1), we have for some $s \in S$

$$h^*(P) = 1 + \max\left\{h^*_P(S \cap I^-_P(s)), h^*_P(S - I_P(s))\right\}. \tag{2}$$

We may restate SPOS as follows: *given a directed graph G_P representing the Hasse diagram of a partial order P, compute an optimal tree for P.*

We close this section with some definitions that will be useful later. Recall that we have a fixed partial order $P = (S, \prec)$ and a fixed set $X \subseteq S$. Let s and t be arbitrary elements of S.

The elements s and t are said to be *comparable* if $s = t$, $s \prec t$, or $t \prec s$, being otherwise said to be *incomparable*; a *chain* in P is a set of pairwise comparable elements and an *antichain* of P is a set of pairwise incomparable elements; the *height* of P, denoted $h(P)$, is the cardinality of a maximum chain in P

and the *width* of P, denoted $w(P)$, is the cardinality of a maximum antichain in P.

A *post* of $P = (S, \prec)$ is an element of S that is comparable to every element of S. We denote the set of posts of P by π_P and note that π_P is a chain in P.

An element s is *maximal* if there is no t such that $s \prec t$. The set of maximal elements in P will be called the *first layer* of P and will be denoted $L_1(P)$. For each $1 < k \leq h(P)$, the *k-th layer* of P, denoted $L_k(P)$, is defined as the first layer of $P - \bigcup_{i=1}^{k-1} L_i(P)$.

A usual, we let $[n] = \{1, \ldots, n\}$ for any integer n. The set of n-element subsets of a given set X will be denoted $\binom{X}{n}$. We write lg for $\log_2$ and log for the natural logarithm.

Recall that an instance to SPOS is a directed graph G_P representing the Hasse diagram of $P = (S, \prec)$. Given an algorithm for SPOS, we focus on the maximal height of the trees computed by this algorithm as a function of the number of elements in the input order P.

We note that when P is a total order (that is, when S is a chain), an optimal tree for P is the usual binary search tree for S. In this case, the height of the corresponding tree is $\lfloor \lg n \rfloor + 1$ and such a tree may be built in polynomial time. On the other extreme, if S is an antichain, the height of the optimal tree is n.

Organization. This paper is organized as follows. In Section 2 we mention some related results concerning searching in partially ordered sets. In Section 3 we show that SPOS is $\mathcal{NP}$-hard, by showing that the corresponding decision problem is $\mathcal{NP}$-complete. In Section 4 we present a polynomial time algorithm for building a tree that has height almost surely bounded by $\lg n + O\left((\log n)^{1/2} \log\log n\right)$ under the random graph model for n-element partial orders. In Section 5 we present a polynomial time algorithm for building a tree that has height almost surely bounded by $6.34 \log_3 n$ under the uniform model for n-element partial orders (assuming the search model with the 'generous' hider).

In Section 6 we make some general remarks and discuss some connections between our results and a related problem studied in [14].

2 Related Work

As mentioned above, we show in Section 3 that the problem of searching through a partially ordered set is $\mathcal{NP}$-hard. We note that our proof may be easily modified so as to encompass the case in which the given order P has a maximum element. The more restricted case in which P has a maximum element and G_P is a tree (the so called *rooted tree* case) may be solved in polynomial time, as proved in [4], where the authors give a $O(n^4(\log n)^3)$ time algorithm for computing an optimal tree (as before, n denotes the number of elements in P). This algorithm does not yield an easy way to estimate the height of the computed tree, except in a few cases; for instance, when G_P is a complete binary tree, the search tree built by their algorithm has height $\lg n + \lg^* n + \Theta(1)$.

A much simpler algorithm for the rooted tree case is presented in [9]. This algorithm computes in time $O(n \log n)$ a search tree whose height exceeds the optimum by at most $\lg n$. Since the optimal tree must have height at least $\lg n$, this algorithm constitutes a 2-approximation algorithm for SPOS for such instances.

Concerning the approximability in the worst case of the general problem, there is a polynomial time $O(\log n)$-approximation algorithm, since one can easily check that SPOS is a restriction of the decision tree problem for which a simple greedy algorithm with such an approximation ratio is known [2]. Optimized exponential time algorithms for building search trees for partial orders are presented in [15]. However, that work considers the minimization of the path length of the tree, instead of its height.

A different although related problem is considered in [14], motivated by the following setting: suppose we are given an $m \times n$ real matrix M whose distinct entries are known to be increasing along the rows and along the columns, and suppose we wish to decide whether a given real number u occurs in M. The goal is to devise a search strategy which minimizes the number of inspections of entries of M in the worst case.

If one looks at the matrix as the product of two chains of length m and n, the problem may be thought of as a problem of searching in a partially ordered set. However, the underlying assumption that the entries of M come from a totally ordered set actually turns it into a different problem, which we discuss in Section 6. The work in [13] and [14] determines bounds for arbitrary orders and studies in detail the case in which P is a product of chains and the case in which G_P is a rooted tree.

3 Computational Complexity

In this section, we prove that SPOS is $\mathcal{NP}$-hard, answering the main question raised in [4]. We shall reduce the exact cover by 3-sets problem, which is $\mathcal{NP}$-complete (see [16, p. 201]), to a decision problem version of SPOS. We start by stating precisely the problems involved in the reduction.

Exact Cover by 3-Sets (EC)
Input: A finite set X of size $3n$ and a family $\Delta \subseteq \binom{X}{3}$.
Question: Is there a set $\Gamma \subseteq \Delta$ with $|\Gamma| = n$ such that $\bigcup_{\gamma\in\Gamma} \gamma = X$?

The decision problem version of SPOS that we consider is as follows.

Search in Partially Ordered Set (SPOSd)
Input: A graph G_P, representing the Hasse diagram of an order P, and an integer k.
Question: Is there a tree for P with height at most k?

We now describe a polynomial reduction from E3C to SPOSd. Let an instance (X, Δ) for E3C be given, where $|X| = 3n$ and $\Delta \subseteq \binom{X}{3}$. We define an instance $(P_{X,\Delta}, k)$ to SPOSd from (X, Δ) as follows. We let $k = |X|/3+2|\Delta|+3$. To define $P_{X,\Delta}$, we first let $\bar{x}_1, \bar{x}_2, \bar{x}_3 \notin X$ be three new elements, and let $\bar{X} = \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$. The partial order $P_{X,\Delta} = (S_{X,\Delta}, \prec_{X,\Delta})$ is defined as follows. The set $S_{X,\Delta}$ is given by

$$S_{X,\Delta} = \Delta \cup X \cup \bar{X} \cup \bar{Q},$$

where

$$\bar{Q} = (X \cup \bar{X}) \times \{1, \ldots, 2|\Delta|\}.$$

The relation $\prec_{X,\Delta}$, which we henceforth denote simply by $\prec$, is the smallest order relation such that $x \prec \delta$ for all $x \in \delta \in \Delta$ and $(x, j) \prec x$ for all $x \in X \cup \bar{X}$ and all $j \in \{1, \ldots, 2|\Delta|\}$ (see Figure 1).

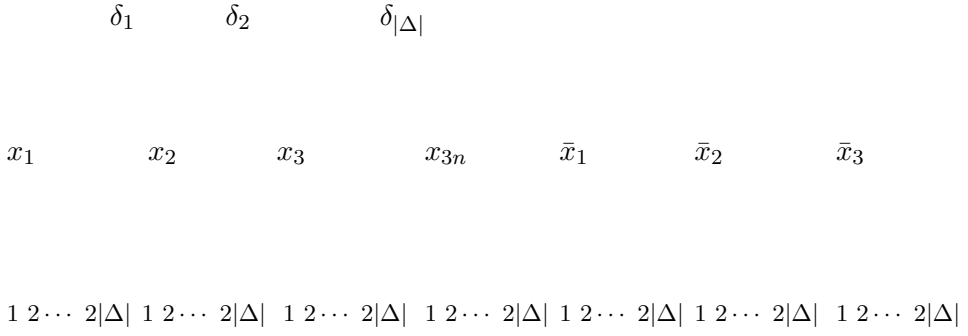

Fig. 1. A schematic view of the order $P_{X,\Delta}$

We shall now prove that there is an exact cover for (X, Δ) if and only if there is a tree for $P_{X,\Delta}$ of height at most $k = |X|/3 + 2|\Delta| + 3$. We start with a simple fact.

Lemma 1 *Let $\delta \in \Delta$ be given. The height of an optimal tree for $I^-(\delta)$ is at most $3 + 2|\Delta|$.*

PROOF. It suffices to present a search strategy for the seeker that makes at most $3 + 2|\Delta|$ queries. Suppose $\delta = \{x_a, x_b, x_c\}$. The seeker first queries whether u, the hider's choice, is x_a, x_b, or x_c. If he gets a hit for any of these queries, or else if he gets three no answers, then he is done. If the answer is smaller for some x_α ($\alpha \in \{a, b, c\}$), then he queries the $2|\Delta|$ elements strictly below x_α in $P_{X,\Delta}$. Clearly, at worst, $3 + 2|\Delta|$ queries will be required.

The above informal description of the search strategy should suffice, but we include a brief description of a binary search tree T that formalizes the strategy. The rightmost path of T has nodes x_a, x_b, and x_c. Furthermore, the left subtree of x_α ($\alpha \in \{a, b, c\}$) is a tree for $I^-(x_\alpha)$ with height $2|\Delta|$. Clearly, $h(T) = 3 + 2|\Delta|$ (see Figure 2). □

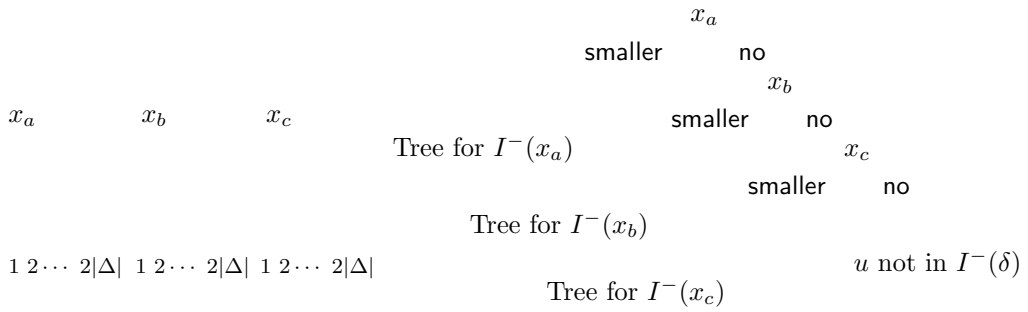

Fig. 2. The order induced by $I^-(\delta)$ and the search tree for $I^-(\delta)$

Lemma 2 *If there is an exact cover for (X, Δ), then there is a tree for $P_{X,\Delta}$ of height at most $k = |X|/3 + 2|\Delta| + 3 = n + 2|\Delta| + 3$.*

PROOF. We suppose that (X, Δ) has an exact cover, and describe a search strategy for the seeker that makes at most $n + 2|\Delta| + 3$ queries. Let $\{\delta_i : 1 \leq i \leq n\} \subseteq \Delta$ be an exact cover for X. The seeker proceeds as follows. He first queries whether u is $\delta_1, \ldots, \delta_n, \bar{x}_1, \bar{x}_2, \bar{x}_3$, in this order. If he gets a hit, or else if he gets $n + 3$ no answers, he is done.

Suppose the answer is smaller for the query δ_i for some $1 \leq i \leq n$. Then he uses the search strategy of Lemma 1, which will require at most $3 + 2|\Delta|$ further queries, giving a total of at most $n + 2|\Delta| + 3$ queries, as required.

Suppose now that the answer is smaller for the query $\bar{x}_j$ for some $1 \leq j \leq 3$. Then with $2|\Delta|$ further queries to the elements strictly below $\bar{x}_j$ in $P_{X,\Delta}$, he will be done. □

Suppose $x \in X \cup \bar{X}$. It is easy to devise a 'hiding strategy' that forces $2|\Delta|+1$ queries of the seeker in a search for an element $u \in I(x)$. The following fact, which will be used in the proof of Lemma 4, is a generalization of this remark.

Fact 3 *Let $x \in X \cup \bar{X}$ and $S' \subseteq S_{X,\Delta}$ be given. If $I(x) \subseteq S'$, then the height of the optimal tree for S' is at least $2|\Delta|+1$.* □

Lemma 2 and Lemma 4 below prove that our reduction from E3C to SPOSd works.

Lemma 4 *If $P_{X,\Delta}$ admits a tree of height at most $k = |X|/3 + 2|\Delta| + 3 = n + 2|\Delta| + 3$, then there is an exact cover for (X, Δ).*

PROOF. Let T be an optimal tree for $P_{X,\Delta}$, and suppose

$$h(T) \leq k = n + 2|\Delta| + 3. \tag{3}$$

The first observation is that the rightmost path R of T has at least $n+3$ nodes. To see this, first observe that each of the $3n+3$ elements in $X \cup \bar{X}$ is either in R, or else is smaller than some element in R. However, an element s in $P_{X,\Delta}$ has below it either at most 3 elements of X or else at most 1 element of $\bar{X}$, that is, $|I(s) \cap X| \leq 3$ and $I(s) \cap \bar{X} = \emptyset$, or else $I(s) \cap X = \emptyset$ and $|I(s) \cap \bar{X}| \leq 1$. Therefore, R must have at least $n+3$ nodes.

Let $a_1, \ldots, a_{n+3}$ be the first $n+3$ nodes in R. More precisely, let a_1 be the root of T, and let a_i be the right child of a_{i-1} $(1 < i \leq n+3)$. We put $A = \{a_1, \ldots, a_{n+3}\}$.

Given $x \in X \cup \bar{X}$, we say that x is *good* if it is *not* comparable in $P_{X,\Delta}$ to any element in A, that is, $x \notin I(A) \cup F(A)$.

We claim that if $x \in X \cup \bar{X}$ is good, then $I(x)$ is contained in the right subtree of a_{n+3}. To see this, fix $y \in I(x)$, and suppose y is *not* in the right subtree of a_{n+3}. Then y is in A, or else it is in the left subtree of some $a_j \in A$. Since x is comparable with y and x is good, we cannot have $y \in A$. Therefore, y is in the left subtree of some a_j, that is, $y \prec a_j$. Since $y \in I(x)$ and $x \in X \cup \bar{X}$, we must have $a_j = x$, or else $a_j = \delta$ for some $\delta \in \Delta$ with $x \prec \delta$. In either case, we have x comparable to some element in A, contradicting the hypothesis that x is good. This contradiction shows that $I(x)$ is indeed contained in the right subtree of a_{n+3}, as claimed.

Suppose now that a good element $x \in X \cup \bar{X}$ exists. The fact that $I(x)$ is contained in the right subtree of a_{n+3} and Fact 3 imply that $h(T) \geq n+2|\Delta|+4$. As we are assuming (3), this shows that a good element does not exist, that is, every element of $X \cup \bar{X}$ is comparable to some element in A. In particular,

$|I(\bar{X}) \cap A| \geq 3$, and hence $|A - I(\bar{X})| \leq n$. Since every element of X is comparable to some element in A and $|X| = 3n$, it is easily seen that $A - I(\bar{X})$ must be an exact cover for X. □

Problem SPOSd is in $\mathcal{NP}$ and E3C is $\mathcal{NP}$-complete (see [16, p. 201]). Lemmas 2 and 4 tell us that $(X, \Delta) \mapsto (P_{X,\Delta}, \prec)$ gives a polynomial time reduction from E3C to SPOSd, and hence we are done.

Theorem 5 SPOSd *is $\mathcal{NP}$-complete.* □

4 The Random Graph Model

In this section, we study the case in which the instances to SPOS are generated according to the random graph model. The *random graph order* probability space, denoted $\mathcal{P}_{n,p}$, is the probability space of all orders $([n], \prec)$ obtained by independently choosing each pair of $\{(i,j) \in [n]^2 \colon i < j\}$ with probability p and taking the transitive closure of the resulting relation. We denote a random element of $\mathcal{P}_{n,p}$ by $P_{n,p}$. The reader is referred to [7] for a detailed discussion of this model. We start with some algorithmic considerations.

4.1 The Algorithm

Let $P = (S, \prec)$ be an order and let us define a *median layer* of P as a layer m of P satisfying

$$\left| \bigcup_{i=1}^{m-1} L_i(P) \right| \leq \frac{1}{2}|S| \quad \text{and} \quad \left| \bigcup_{i=m+1}^{h(P)} L_i(P) \right| \leq \frac{1}{2}|S|.$$

Consider the following version of the usual binary search strategy, adapted to the case of partial orders.

Algorithm $\bar{\mathcal{B}}(P)$
(1) $m \leftarrow$ index of a median layer of P;
(2) $L_\uparrow \leftarrow \bigcup_{i=1}^{m-1} L_i(P)$;
(3) $L_\downarrow \leftarrow \bigcup_{i=m+1}^{h(P)} L_i(P)$;
(4) Perform a query about u to each $s \in L_m(P)$:
  (a) if the outcome of one of these queries is hit, return yes;
  (b) if the outcome of one of these queries is smaller, return $\bar{\mathcal{B}}(P(L_\downarrow))$;
  (c) if the outcome to all these queries is no, return $\bar{\mathcal{B}}(P(L_\uparrow))$;

Let us call the above strategy the *extended binary search*. We write $\mathcal{B}$ for the algorithm that, given a directed graph G_P representing the Hasse diagram of P, produces the binary search tree corresponding to the algorithm $\bar{\mathcal{B}}(P)$ given above. Since the decomposition of P into its layers can be computed in polynomial time using breadth-first search, we clearly have the following fact.

Fact 6 *Algorithm $\mathcal{B}$ computes in polynomial time a binary search tree for P of height at most $w(P)(\lfloor \lg h(P) \rfloor + 1)$.* □

We now turn our attention to another strategy: If $\pi_P \neq \emptyset$, then let $d_1 \prec d_2 \prec \ldots \prec d_k$ ($k = |\pi_P|$) be the posts of P and define the *segments* $S_i(P)$ *of* P by

$$S_i(P) = \begin{cases} I(d_1), & \text{if } i = 0; \\ I(d_{i+1}) - I(d_i), & \text{if } 0 < i < k; \\ S - I(d_k), & \text{if } i = k. \end{cases} \tag{4}$$

We now consider the following search strategy.

Algorithm $\bar{\mathcal{A}}(P)$
If $\pi_P = \emptyset$, return $\bar{\mathcal{B}}(P)$;
otherwise perform the usual binary search on π_P, looking for u.
If u is found, return yes;
otherwise, return $\bar{\mathcal{B}}(P(S_{i-1}(P)))$,
where $i = \min\Big(\{j\colon u \prec d_j\} \cup \{k+1\}\Big)$.

Let $\mathcal{A}$ be the algorithm that, given a directed graph G_P representing the Hasse diagram of P, produces the binary search tree corresponding to the algorithm $\bar{\mathcal{A}}(P)$ given above.

Fact 7 *If $\pi_P \neq \emptyset$, algorithm $\mathcal{A}$ computes in polynomial time a binary search tree for P of height at most* $\lg n + 1 + w(P)\Big(\lfloor \lg \max_{0 \leq i \leq |\pi_P|} |S_i(P)| \rfloor + 1\Big)$. □

4.2 Analysis

In this section, we shall show that the algorithm $\mathcal{A}$ from Section 4.1 performs well when run on input $P_{n,p}$, *if p is a constant*. In fact, this will follow easily from the fact that, for p constant, a typical $P_{n,p}$ is such that $w(P_{n,p}) = O(\sqrt{\log n})$ and, for all segments $S_i(P_{n,p})$ of $P_{n,p}$, we have $|S_i(P_{n,p})| = O(\log n)$. The analysis in Section 4.1 will therefore imply that the height of the tree computed by $\mathcal{A}$ is at most $\lg n + O((\log n)^{1/2} \log\log n)$.

Let us first consider the cardinality of the segments of $P_{n,p}$. Recall that the

segments of an order are defined by its posts. The notion of post is crucial in the investigation of the structure of $P_{n,p}$ (see, for instance, [1,6]); here, it will suffice to make use of some basic facts about them.

One may define the random graph order $P_{\mathbb{Z},p} = (\mathbb{Z}, \prec)$ on $\mathbb{Z}$ in the same way that we defined $P_{n,p} = ([n], \prec)$ on $[n]$. It turns out that the distances between consecutive posts of $P_{\mathbb{Z},p}$ are independent, identically distributed random variables [1]. Let us write L for a random variable with this distribution. Recently, settling a question raised in [1], the following tail inequality for L has been proved.

Theorem 8 (Kim and Pittel [11]) *Let $0 < p < 1$ be fixed. There exists a constant $c = c(p) > 0$ such that $\mathbb{P}(L > l) \leq \exp(-cl)$ for all $l > 0$.* □

An immediate consequence of the above result is as follows.

Corollary 9 *Let $0 < p < 1$ be fixed and let C be a constant with $C > 1/c$, where $c = c(p)$ is as in Theorem 8. The random graph order $P_{n,p}$ has almost surely no segment with more than $C \log n$ elements.*

PROOF. The cardinality of the segment $S_i(P_{n,p})$ $(1 \leq i < k = |\pi_{n,p}|)$ of $P_{n,p}$ is the distance $d_{i+1} - d_i$ between the posts d_i and d_{i+1} (we follow the notation introduced in Section 4.1; see (4)). Moreover, the segments $S_0(P_{n,p})$ and $S_k(P_{n,p})$ have cardinalities d_1 and $n - d_k$.

It follows from Theorem 8 that the random variables $|S_i(P_{n,p})|$ $(0 \leq i \leq k)$ satisfy the exponential tail inequality in that result. In particular, for any fixed i, the probability that $|S_i(P_{n,p})| > C \log n$ is $o(1/n)$. As $P_{n,p}$ has at most n segments, the probability that some segment $|S_i(P_{n,p})|$ has cardinality greater than $C \log n$ is $o(1)$. □

Consider now a layer $L_i(P_{n,p})$ of $P_{n,p}$. Since $L_i(P_{n,p})$ is an antichain of $P_{n,p}$, we may make direct use of the following result, proved by Barak and Erdős [3]. We follow Bollobás and Brightwell [5], where an oversight in [3] is corrected.

Theorem 10 *Let p be a constant with $0 < p < 1$ and set $q = 1 - p$. Let $\delta > 0$ be a constant and set*

$$K_{n,p} = \sqrt{\frac{2 \log n}{\log(1/q)} + \frac{1}{4}} + \frac{1}{2}, \tag{5}$$

so that $nq^{\binom{K_{n,p}}{2}} = 1$. Then, almost surely, the width $w(P_{n,p})$ of $P_{n,p}$ satisfies

$$\lfloor K_n - \delta \rfloor \leq w(P_{n,p}) < \lceil K_n + \delta \rceil.$$ □

We are now in position to prove the main result of this section, namely, we shall now show that, almost surely, the algorithm $\bar{\mathcal{A}}(P_{n,p})$ from Section 4.1 makes at most $\lg n + O(\sqrt{\log n} \log\log n)$ queries in the worst case.

Theorem 11 *Let $0 < p < 1$ be fixed. Almost surely, the search tree corresponding to the algorithm $\bar{\mathcal{A}}(P_{n,p})$ has height at most $\lg n + O(\sqrt{\log n} \log\log n)$.*

PROOF. In view of Corollary 9 and Theorem 10, there are constants $c_1 = c_1(p)$ and $c_2 = c_2(p)$ that depend only on p such that, almost surely, $P_{n,p}$ satisfies $w(P_{n,p}) \leq c_1\sqrt{\log n}$ and all segments $S_i(P_{n,p})$ of $P_{n,p}$ have cardinality at most $c_2 \log n$. Recalling Fact 7, Theorem 11 follows. □

5 The Uniform Model

In this section we study the problem of searching in a typical partial order according to the uniform model. We start by stating some definitions and a key auxiliary result.

Denote by $\mathcal{P}(n)$ the set of all partial orders on $[n]$. Taking $\mathcal{P}(n)$ with the uniform distribution, that is, making each partial order equally likely, we have the *uniform model* for random partial orders; a random element in this model will be denoted U_n. The reader is again referred to [7] for a detailed discussion of this model.

It is known that almost all U_n have a strong structural property, which we now describe. Let (X_1, X_2, X_3) be a partition of $[n]$, and let $\mathcal{A}(X_1, X_2, X_3)$ be the set of partial orders $P = ([n], \prec)$ such that every $x_3 \in X_3$ is smaller than every $x_1 \in X_1$, and such that if $x_i \in X_i$, $x_j \in X_j$, and $x_i \prec x_j$, then $i > j$. Thus, an order P in $\mathcal{A}(X_1, X_2, X_3)$ is determined by arbitrarily selecting to be in P some relations of the forms $x_3 \prec x_2$ and $x_2 \prec x_1$ ($x_i \in X_i$, $1 \leq i \leq 3$). In particular, $|\mathcal{A}(X_1, X_2, X_3)| = 2^{|X_1||X_2|+|X_2||X_3|}$.

Answering the question "*what does a 'typical' partial order on $[n]$ look like?*", Kleitman and Rothschild proved the following rather surprising result.

Theorem 12 (Kleitman and Rothschild [12]) *Let $\omega = \omega(n)$ be an arbitrary function such that $\omega \to \infty$ as $n \to \infty$. Almost every partial order on $[n]$ lies in $\mathcal{A}(X_1, X_2, X_3)$ for some partition (X_1, X_2, X_3) of $[n]$ with $\Big||X_2| - n/2\Big| < \omega$ and $\Big||X_3| - n/4\Big| < \omega\sqrt{n}$.* □

Theorem 12 makes the problem of searching in typical partial orders as stated in Section 1 rather uninteresting, since our search model makes it unavoidable

to query each of the maximal elements of the given order, and Theorem 12 tells us that almost all orders have $(1/4 + o(1))n$ such elements.

To make the problem more interesting, we now consider a variant of our search model, where a query to s about u has four possible outcomes: smaller, greater, hit and no meaning, respectively, $u \prec s$, $s \prec u$, $s = u$ and "*s is not comparable to u*"; accordingly, a strategy is redefined to be a *ternary* search tree (see Section 1.1). In this section we shall prove that, under the uniform model, it is almost always possible to construct a ternary search tree of height $O(\log n)$ in polynomial time.

5.1 Probabilistic Preliminaries

For the remainder of Section 5, an arbitrary function $\omega = \omega(n)$ with $\omega \to \infty$ as $n \to \infty$ is fixed. For convenience, we let $\Pi_n(\omega)$ denote the set of partitions (X_1, X_2, X_3) of $[n]$ as in the statement of Theorem 12, that is, such that $\big||X_2| - n/2\big| < \omega$ and $\big||X_3| - n/4\big| < \omega\sqrt{n}$. Also, let

$$\mathcal{A}_{\mathrm{KR}}(n, \omega) = \bigcup_{\Pi_n(\omega)} \mathcal{A}(X_1, X_2, X_3), \tag{6}$$

where the union is taken over all $(X_1, X_2, X_3) \in \Pi_n(\omega)$. Note that Theorem 12 asserts that $|\mathcal{A}_{\mathrm{KR}}(n, \omega)|/|\mathcal{P}(n)| \to 1$ as $n \to \infty$.

A standard argument, given below for completeness, allows us to restrict our attention to the spaces $\mathcal{A}(X_1, X_2, X_3)$ ($(X_1, X_2, X_3) \in \Pi_n(\omega)$) when proving that a given property happens almost surely in the probability space $\mathcal{U}_n$.

Lemma 13 *Let $\mathcal{E} \subseteq \mathcal{P}(n)$ be an event such that the probability that $U_n \in \mathcal{E}$, conditional on $U_n \in \mathcal{A}(X_1, X_2, X_3)$, tends to 1 as $n \to \infty$ uniformly on $(X_1, X_2, X_3) \in \Pi_n(\omega)$. Then $U_n \in \mathcal{E}$ almost surely.*

PROOF. Let $\mathcal{E}^{\mathrm{c}} = \mathcal{P}(n) - \mathcal{E}$ and $\mathcal{A}_{\mathrm{KR}}(n, \omega)^{\mathrm{c}} = \mathcal{P}(n) - \mathcal{A}_{\mathrm{KR}}(n, \omega)$. We use the hypothesis on $\mathcal{E}$ and Theorem 12 to observe that, in $\mathcal{U}_n$, we have

$$\begin{aligned}\mathbb{P}(\mathcal{E}^{\mathrm{c}}) &= \mathbb{P}(\mathcal{E}^{\mathrm{c}} \cap \mathcal{A}_{\mathrm{KR}}(n, \omega)) + \mathbb{P}(\mathcal{E}^{\mathrm{c}} \cap \mathcal{A}_{\mathrm{KR}}(n, \omega)^{\mathrm{c}}) \\ &\le \sum_{\Pi_n(\omega)} \mathbb{P}(\mathcal{E}^{\mathrm{c}} \mid \mathcal{A}(X_1, X_2, X_3))\mathbb{P}(\mathcal{A}(X_1, X_2, X_3)) + \mathbb{P}(\mathcal{A}_{\mathrm{KR}}(n, \omega)^{\mathrm{c}}) \\ &= o(1) \sum_{\Pi_n(\omega)} \mathbb{P}(\mathcal{A}(X_1, X_2, X_3)) + o(1) = o(1),\end{aligned}$$

where the sums above are over all $(X_1, X_2, X_3) \in \Pi_n(\omega)$ (see (6)). □

The binomial bipartite order. Let X and Y be two disjoint sets and let $\mathcal{A}(X,Y)$ be the family of orders P on $X \cup Y$ defined by putting each $(y,x) \in Y \times X$ in P independently, with probability $1/2$. In particular, for any fixed $Q \subseteq X$, if N is the number of elements $y \in Y$ with a given comparability/incomparability relation with all $x \in Q$, then its expectation is

$$\mathbb{E}[N] = |Y|2^{-|Q|}. \tag{7}$$

It follows from the definition of the space $\mathcal{A}(X,Y)$ that all the $2^{|X||Y|}$ bipartite orders in $\mathcal{A}(X,Y)$ are equiprobable. Furthermore, if (X_1,X_2,X_3) is a partition of $[n]$, then there is a natural bijection between $\mathcal{A}(X_1,X_2,X_3)$ and $\mathcal{A}(X_1,X_2)\times\mathcal{A}(X_2,X_3)$. Since all the above spaces are uniform (all orders are equiprobable), this bijection shows that we may identify the probability spaces $\mathcal{A}(X_1,X_2,X_3)$ and $\mathcal{A}(X_1,X_2)\times\mathcal{A}(X_2,X_3)$ in the obvious way.

5.2 Searching 3-Layered Orders

Let $P \in \mathcal{P}(n)$ be an order, let L be a layer of P, let S be the union of the layers adjacent to L, and let $Q \subseteq L$. The following simple algorithm decides whether $u \in S$:

Algorithm $\bar{\mathcal{C}}(Q,S)$

(1) For each $q \in Q$, query q about u:
  if the answer is hit, return no [a];
  if the answer is smaller, let $S \leftarrow S \cap I(q)$;
  if the answer is greater, let $S \leftarrow S \cap F(q)$;
  if the answer is no, let $S \leftarrow S - (I(q) \cup F(q))$.

(2) For each $s \in S$,
  query s about u and, if the answer is hit, return yes.

(3) Return no.

---

[a] Recall $Q \cap S = \emptyset$.

The number of queries $c(Q,S)$ made by $\bar{\mathcal{C}}(Q,S)$ is clearly at most $|Q| + |r(Q,S)|$, where $r(Q,S)$ is the set of elements still remaining in S at the beginning of line 2.

We now consider the case in which $\bar{\mathcal{C}}(Q,S)$ is run on $P \in \mathcal{A}(X_1,X_2,X_3)$, where $(X_1,X_2,X_3) \in \Pi_n(\omega)$. In what follows, we fix a function $\omega' = \omega'(n) = o(\log n)$ with $\omega' \to \infty$ as $n \to \infty$.

Lemma 14 *Let* $q(n) = \lceil 2\lg n + \omega' \rceil$*, and suppose* $(X_1,X_2,X_3) \in \Pi_n(\omega)$ *and* $P \in$

$\mathcal{A}(X_1, X_2, X_3)$. *If we run* $\bar{\mathcal{C}}(Q, S)$ *on* P, *where* $Q \subseteq X_2$ *has cardinality* $|Q| = q(n)$ *and* $S = X_1 \cup X_3$, *then almost surely we have* $c(Q, S) \leq q(n)+1$. *Also, if we run* $\bar{\mathcal{C}}(Q, S)$ *on* P, *where* $Q \subseteq X_1$ *has cardinality* $|Q| = q(n)$ *and* $S = X_2$, *then* $c(Q, S) \leq q(n) + 1$.

PROOF. Both assertions follow from the fact that $q(n)$ queries almost surely 'separate' all pairs in S. Let Z be the number of pairs $(x, y) \in S \times S$, $x \neq y$, such that $\{x, y\} \subset r(Q, S)$. The expectation of Z is easily seen to be at most $|S|^2 2^{-|Q|} \leq n^2 2^{-q(n)} = o(1)$ (recall (7)). Therefore, $Z = 0$ with probability $1 - o(1)$, and hence almost surely $|r(Q, S)| \leq 1$. □

We now consider the following search strategy $\bar{\mathcal{D}}(P)$, which works well when $P \in \mathcal{A}(X_1, X_2, X_3)$ and $(X_1, X_2, X_3) \in \Pi_n(\omega)$. Intuitively, $\bar{\mathcal{D}}(P)$ first selects $q(n)$ elements from X_2 and queries them; this will locate u if u is in $X_1 \cup X_3$. If $u \notin X_1 \cup X_3$, then $\bar{\mathcal{D}}(P)$ tries to locate u in X_2; to do so, $\bar{\mathcal{D}}(P)$ queries $q(n)$ elements from X_1.

Algorithm $\bar{\mathcal{D}}(P)$

Let n be the number of elements in P and let $q(n) = \lceil 2 \lg n + \omega' \rceil$.

(1) If $h(P) \neq 3$, $|L_2(P)| < q(n)$, or $|L_1(P)| < q(n)$, return $\bar{\mathcal{B}}(P)$ [a].
(2) $S \leftarrow L_1(P) \cup L_3(P)$; $Q \leftarrow$ a subset of $L_2(P)$ of size $q(n)$
(3) if $\bar{\mathcal{C}}(Q, S) =$ yes, then return yes.
(4) $S \leftarrow L_2(P)$; $Q \leftarrow$ a subset of $L_1(P)$ of size $q(n)$
(5) return $\bar{\mathcal{C}}(Q, S)$.

---

[a] See Section 4 for $\bar{\mathcal{B}}(P)$.

Suppose $P \in \mathcal{A}(X_1, X_2, X_3)$ and $(X_1, X_2, X_3) \in \Pi_n(\omega)$. Then, by Lemma 14, almost surely the number of queries made by $\bar{\mathcal{D}}(P)$ is at most $2(q(n) + 1) = (4 + o(1)) \lg n = (6.33\ldots + o(1)) \log_3 n$.

Theorem 15 *Almost every* U_n *admits a ternary search tree of height at most* $6.34 \log_3 n$, *and such a tree may be constructed in polynomial time.* □

6 Concluding Remarks

We remark that our reduction in Section 3 resembles the one employed in [10] to prove that the optimal decision tree problem is $\mathcal{NP}$-complete. In fact, SPOS may be viewed as a restriction of the optimal decision tree problem.

In Section 4, we consider only the case in which p is a constant, independent of n. It would be of interest to investigate the case in which $p \to 0$ as $n \to \infty$. Results similar to the ones in Section 4 hold if p tends to 0 very slowly, but a completely new approach will be required to deal with the case in which, say, $p \ll 1/\log n$ (see [5] and [6]).

We observe that one may also study the uniform model conditioning on having sparser partial orders. To carry out this investigation, the recent results in [17] and [18] would be the starting point. We also refer the reader to [8] for a remarkable sharpening of Kleitman and Rothschild's theorem.

As mentioned in Section 2, Linial and Saks [14] consider a different, although related problem where the set U is totally ordered, the given partial order $P = (S, \prec)$ is 'compatible' with this total order (the total order is a linear extension of $\prec$), and where each query about $u \in U$ to $s \in S$ is made *with respect to the total order* and not with respect to P, as is our case.

To see why this turns out to define a different problem, consider what information is gained in a search through S for $u \in U$ when we query $s \in S$ about u and the outcome is smaller: in our problem, such an outcome is enough to confine the remaining of the search to $S \cap I(s)$; in their problem, however, this is not the case: as the query is made with respect to the total order, the outcome smaller leaves all elements in $S - \{x \in S \colon s \prec x\}$ as valid candidates.

While not presenting explicitly an algorithm to compute an optimal tree for their problem, it is a consequence of the results in [13] and [14] that the height H of an optimal tree for the problem satisfies

$$\lg \iota(P) \le H \le \kappa \lg \iota(P), \tag{8}$$

where $\iota(P)$ is the number of ideals of P and $\kappa = (2 - \lg(1 + \lg 5))^{-1} = 3.73\ldots$. We note that this fact alone leads to results similar to those reached in Section 4 when we consider *their* problem for random graph orders with p constant. Let us briefly discuss this.

A possible search strategy is again to isolate one segment by means of a binary search restricted to the posts of the order and then to search through this segment using an extended binary search (algorithm $\bar{\mathcal{B}}$).

An ideal of a partial order is uniquely determined by the antichain of its maximal elements. Therefore, the number of ideals in a segment R is bounded by the number of antichains it contains, and hence

$$\iota(P(R)) \le \sum_{i=1}^{w(P)} \binom{|R|}{i} \le |R|^{w(P)}.$$

Together with the bounds in (8) and Theorem 10, this allows us to deduce

the existence of a tree for R of height at most $\kappa \lg \iota(P(R)) = O(w(P) \lg |R|)$. Hence, for any partial order, there is a tree of height $H \leq \lg n + O(w(P) \lg r)$, where r is the maximal cardinality of the segments of P. Again, by Corollary 9 and Theorem 10, we almost surely have $H \leq \lg n + O((\log n)^{1/2} \log \log n)$.

Acknowledgements. The authors are much indebted to the referees for their thorough and most valuable comments and suggestions. In particular, a referee simplified the reduction that we originally had for the proof of the $\mathcal{NP}$-completeness of SPOSd, suggested improvements to the algorithms in Section 5, and also told us about [11], which simplified the exposition in Section 4.

References

[1] N. Alon, B. Bollobás, G. Brightwell, and S. Janson. Linear extensions of a random partial order. *Ann. Appl. Probab.*, 4:108–123, 1994.

[2] E. Arkin, H. Meijer, J. Mitchell, D. Rappaport, and S. Skiena. Decision trees for geometric objects. *Internat. J. Comput. Geom. Appl.*, 8:343–363, 1998.

[3] A. B. Barak and P. Erdős. On the maximal number of strongly independent vertices in a random acyclic directed graph. *SIAM Journal on Algebraic and Discrete Methods*, 5(4):508–514, 1984.

[4] Y. Ben-Asher, E. Farchi, and I. Newman. Optimal search in trees. *SIAM Journal on Computing*, 28(6):2090–2102, Dec. 1999.

[5] B. Bollobás and G. Brightwell. The width of random graph orders. *Math. Sci.*, 20(2):69–90, 1995.

[6] B. Bollobás and G. Brightwell. The structure of random graph orders. *SIAM Journal on Discrete Mathematics*, 10(2):318–335, May 1997.

[7] G. Brightwell. Models of random partial orders. In *Surveys in combinatorics, 1993 (Keele)*, volume 187 of *London Math. Soc. Lecture Note Ser.*, pages 53–83. Cambridge Univ. Press, Cambridge, 1993.

[8] G. Brightwell, H. J. Prömel, and A. Steger. The average number of linear extensions of a partial order. *J. Combin. Theory Ser. A*, 73(2):193–206, 1996.

[9] L. G. Holanda. Algoritmos e novos limites para busca em conjuntos parcialmente ordenados. Master's thesis, COPPE-UFRJ, 2001. (In Portuguese).

[10] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, May 1976.

[11] J. H. Kim and B. Pittel. On tail distribution of interpost distance. *Journal of Combinatorial Theory Series B*, 80:49–56, 2000.

[12] D. J. Kleitman and B. L. Rothschild. Asymptotic enumeration of partial orders on a finite set. *Trans. Amer. Math. Soc.*, 205:205–220, 1975.

[13] N. Linial and M. Saks. Every poset has a central element. *Journal of Combinatorial Theory, Series A*, 40:195–210, 1985.

[14] N. Linial and M. Saks. Searching ordered structures. *Journal of Algorithms*, 6(1):86–103, Mar. 1985.

[15] M. J. Lipman and J. Abrahams. Minimum average cost testing for partially ordered components. *IEEE Transactions on Information Theory*, 41(1):287–291, Jan 1995.

[16] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[17] H. J. Prömel, A. Steger, and A. Taraz. Counting partial orders with a fixed number of comparable pairs. *Combin. Probab. Comput.*, 10(2):159–177, 2001.

[18] H. J. Prömel, A. Steger, and A. Taraz. Phase transitions in the evolution of partial orders. *J. Combin. Theory Ser. A*, 94(2):230–275, 2001.