Searching in Random Partially Ordered Sets
(Extended Abstract)

R. Carmo[1,2]*, J. Donadelli[2]**, Y. Kohayakawa[2]***, and E. Laber[3]

[1] Departamento de Informática
Universidade Federal do Paraná
Centro Politécnico da UFPR, 81531–990, Curitiba PR, Brazil
`http://www.inf.ufpr.br`
`renato@inf.ufpr.br`

[2] Instituto de Matemática e Estatística
Universidade de São Paulo
Rua do Matão 1010, 05508–900 São Paulo SP, Brazil
`http://www.ime.usp.br/mac`
{`renato,jair,yoshi`}`@ime.usp.br`

[3] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente 225, Rio de Janeiro RJ, Brazil
`http://www.inf.puc-rio.br/`
`laber@info.puc-rio.br`

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. We consider two classical models for random partial orders, the *random graph model* and the *uniform model*.

We shall show that certain simple, fast algorithms are able to produce nearly-optimal search strategies for typical partial orders under the two models of random partial orders that we consider. For instance, our algorithm for the random graph model produces, in linear time, a search strategy that makes $O\left((\log n)^{1/2}\log\log n\right)$ more queries than the optimal strategy, for almost all partial orders on n elements. Since we need to make at least $\lg n = \log_2 n$ queries for *any* n-element partial order, our result tells us that one may efficiently devise near-optimal search strategies for almost all partial orders in this model (the problem of determining an optimal strategy is NP-hard, as proved recently in [1]).

* Partially supported by PICDT/CAPES.
** Supported by a CNPq doctorate Studentship (Proc. 141633/1998-0).
*** Partially supported by MCT/CNPq through ProNEx Programme (Proc. CNPq 664107/1997–4) and by CNPq (Proc. 300334/93–1 and 468516/2000–0).

1 Introduction

A fundamental problem in data structures is the problem of representing a dynamic set S in such a way that we may efficiently search for arbitrary elements u in S. Perhaps the most common assumption about S is that its elements belong to some *totally ordered set* U. Here, 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*. In fact, we address the basic 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*.

The problem we consider is, intuitively speaking, as follows: suppose we are given a partial order $S = (S, \prec)$, where $S \subset U$ and $\prec$ is a partial order on U. We wish to construct a search strategy T that, given an arbitrary $u \in U$, determines whether or not $u \in S$. We suppose that T has access to an oracle that, given $s \in S$, replies whether or not $s = u$, and if this is not the case then tells T whether or not $u \prec s$. We measure the efficiency of T by the number of queries that it needs to send to the oracle in the worst case (we maximize over all $u \in U$). Our problem may then be summarized as follows:

1. an instance is a partial order $(S, \prec)$, its size given by the size of a directed acyclic graph representing its Hasse diagram;
2. a solution is a search strategy T;
3. the aim is to minimize number of queries in the worst-case performance of T, and
4. the number of steps needed to compute T is to be kept as low as possible.

We shall show that certain simple, fast algorithms are able to produce nearly-optimal search strategies for typical instances under the two models of random partial orders.

1. *Random graph model.* In the *random graph model*, we randomly select a graph on the vertex set $[n] = \{1, \dots, n\}$ and say that $i \prec j$ in our partial order if $i < j$ and one may reach j from i along an 'increasing path' in the graph. (We in fact consider random graphs of density p, for constant p.) We present an algorithm that, *in time linear in the size of the instance* $(S, \prec)$, produces a search strategy that makes at most
$$\lg n + O\left((\log n)^{1/2} \log\log n\right)$$
queries in the worst case, for almost all n-element partial orders in this model (that is, with probability tending to 1 as $n \to \infty$).
2. *Uniform model.* In the *uniform model*, we simply consider all partial orders on $[n]$ equiprobable. The situation here is somewhat different: it is easy to show that almost all n-element partial orders are such that any search strategy makes at least
$$\left(\frac{1}{4} + o(1)\right) n$$

queries in the worst case. However, if we consider a slightly more 'generous' oracle, then almost all partial orders admit search strategies that require only

$$O(\log n)$$

queries. Moreover, this search strategy may be computed in time $O(n^2)$. For this result, we suppose that, presented with the query $s \in S$, the oracle replies whether or not $s = u$, and if this is not the case then it tells us whether $u \prec s$, $s \prec u$, or u is not comparable with s. Thus, our more generous oracle tells us exactly which of the three possibilities holds when $u \neq s$, whereas the previous oracle only told us whether or not $u \prec s$ holds.

Since we need to make at least $\lg n$ queries for *any* n-element partial order, our result tells us that one may efficiently devise near-optimal search strategies for almost all partial orders in the first model, and we may devise search strategies for almost all partial orders in the second model that are only worse than the optimal by a constant factor. We remark that the constants hidden in the big-O notation and in the discussion above are not at all large.

Summarizing, (i) one may compute an essentially best possible search strategy for almost all random graph orders and (ii) one may compute a constant factor approximation for almost all partial orders. This is in pleasant constrast to the fact that determining an optimal strategy is NP-hard, as recently proved in [1]. Finally, as the reader will see, it will be quite surprisingly simple to prove our results.

1.1 Definitions and Notation

A *partial order* is a pair $(U, \prec)$ where U is a set and $\prec$ a binary relation on U which is anti-symmetric, transitive and irreflexive. If x, $y \in U$ then $x \prec y$ stands for $(x, y) \in \prec$. The elements $u, v \in U$ are said to be *comparable* in $(U, \prec)$ if $u \prec v$ or $v \prec u$, being otherwise said to be *incomparable*; a *chain* in $(U, \prec)$ is a set $X \subseteq U$ of pairwise comparable elements; an *antichain* of $(U, \prec)$ is a set $X \subseteq U$ of pairwise incomparable elements; the *height* of $(U, \prec)$ is the size of a maximum chain and the *width* is the size of a maximum antichain. An element $u \in U$ is maximal in $(U, \prec)$ if there is no $x \in U$ such that $u \prec x$.

An *ideal* of $(U, \prec)$ is a set $I \subseteq U$ such that $u \prec i \Rightarrow u \in I$ for all $i \in I$ and a *filter* of $(U, \prec)$ is a set $F \subseteq U$ such that $f \prec u \Rightarrow u \in F$ for all $f \in F$; if $X \in U$, we denote by $I(X)$ (resp. $F(X)$) the minimal ideal I (resp. filter F) of $(U, \prec)$ such $X \subseteq I$ (resp. $X \subseteq F$).

Thoughout the text, we denote a set of integers $\{z \in \mathbb{N}\colon x \leq z \leq y\}$ by $[x, y]$ and define $[n] = [1, n]$ for any integer $n \geq 1$. Also, ω will denote a function $\omega : \mathbb{N} \to \mathbb{R}$ satisfying $\lim_{n\to\infty} \omega(n) = \infty$.

The problem of searching through a partially ordered set can be stated as follows. We are given a partially ordered set $(U, \prec)$ and a finite set $S \subseteq U$ with the partial order induced by $\prec$. Our goal is to determine whether a given $u \in U$ is an element of S and we are allowed to pose *queries* about u to the elements of S.

A *query about $u \in U$ to $s \in S$* has three possible outcomes, namely, hit, smaller and no, meaning $u = s$, $u \prec s$ and $u \not\prec s$, respectively. A *search for $u \in U$ through S* is a sequence of queries allowing to decide whether $u \in S$ or not. The goal is to devise a strategy for querying the elements of S in such a way that the longest search through S poses the smallest number of queries.

Such a strategy can be conveniently thought of as a binary decision tree whose internal nodes are labelled with elements of S, whose external nodes are labelled with pairs $(s_{\min}, s_{\max}) \in (S \cup \{-\infty\}) \times (S \cup \{+\infty\})$ and whose edges are labelled with $\prec$ and $\not\prec$.

A path in the decision tree from its root to an internal node labelled s represents a successful search whose outcome is $u = s$, while a path from the root to an external node labelled $(s_{\min}, s_{\max})$ represents an unsuccessful search whose outcome is $u \notin S$ with the additional information that $s_{\min} \prec u \prec s_{\max}$. We define the *height of a binary decision tree* as the length of a longest path from its root to an external node.

We can then restate our problem as follows: *given a partial order $(S, \prec)$, compute a binary decision tree of minimum height for S.*

We will denote by n the number of elements of S and will assume that the partial order $(S, \prec)$ is given as a directed acyclic graph $G_\prec$ of m edges and n vertices , representing its Hasse diagram. Then, given an algorithm $\mathcal{A}$ for computing such a decision tree, we focus on the height $H_\mathcal{A}(n, m)$ of the tree that we construct and on the number of steps $T_\mathcal{A}(n, m)$ required by $\mathcal{A}$ to construct such tree.

We note that when $\prec$ is a total order on S, the optimum decision tree for S is the usual binary search tree for S. In this case we have $m = n - 1$, $H(n, m) = \lceil \lg n \rceil + 1$ and $T(n, m) = O(n)$, these being the best possible values for H and T.

On the other extreme, if the whole of S is an antichain, then we have $m = 0$, $H(n, m) = n$ and $T(n, m) = O(n)$.

1.2 Organization

This extended abstract is organized as follows. In Section 2 we mention some relevant results in the context of searching in partially ordered sets. In Section 3 we present a linear time algorithm for building a decision tree which 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 4 we present a $O(n^2 \log n)$ time algorithm for building a decision tree which has height almost surely bounded by $O(\log n)$ under the uniform model for n-element partial orders (assuming the search model with the 'generous' oracle).

In Section 5 we make some general ramarks and briefly discuss some connections among our results and a related problem proposed in [2].

2 Related Work

It is shown in [1] that the problem of searching through a partially ordered set is NP-hard, even when restricted to the case in which $(S, \prec)$ has a maximum element. The more restricted case in which the given partial order has a maximum element and $G_\prec$ is a tree (the so called *rooted tree* variant) can be solved in polynomial time as shown in [3], where an algorithm for computing a minimum height decision tree is presented with $T(n,m) = O(n^4(\log n)^3)$. Their algorithm does not yield an easy way to estimate $H(n,m)$, except in a few cases: for instance, when $G_\prec$ is a complete binary tree, the decision tree built by their algorithm has $H(n,m) = \lg n + \lg^* n + \Theta(1)$.

The work in [1] presents a much simpler algorithm for the case in which $G_\prec$ is a rooted tree, which computes in time $T(n,m) = O(n \log n)$ a decision tree whose height is at most $\lg n$ greater than the height of the optimum decision tree. Since the optimum decision tree must have height at least $\lg n$, their algorithm constitutes a 2-approximation algorithm for the problem.

The case in which $(S, \prec)$ has a maximum element and the maximum degree of $G_\prec$ is d is also studied in [1], where it is shown that $d \log_d n$ is an upper bound for the height of an optimum decision tree, which improves to the best possible a previous bound from [4].

Lipman and Abrahams [5] present optimized exponential time algorithms for building decision trees for searching in general partially ordered sets. Nevertheless, they considered the minimization of the expected path length of the decision tree.

Linial and Saks consider in [2] a different although related problem, motivated by the following setting: suppose we are given an $m \times n$ real matrix M whose (say, 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 x 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 5.

Linial and Saks (see [2] and [6]) have determined bounds for arbitrary orders $\prec$ and have studied in detail the case in which $\prec$ is a product of chains and the case in which $G_\prec$ is a rooted tree.

3 The Random Graph Model

Let $(S, \prec)$ be a partial order and denote by $\max_\prec X$ the set of maximal elements of $X \subseteq S$. Consider the following recursive decomposition of S into antichains: let $L_1 = \max_\prec S$, and let $L_i = \max_\prec \left(S - \bigcup_{j=1}^{i-1} L_j\right)$ for $i > 1$. Let h be the height of $(S, \prec)$ and let $w = \max_{1 \le i \le h} |L_i|$. We will call each set L_i the *layer i* of S.

A possible adaptation of the usual binary search strategy to the case of partial orders may be described as follows.

Algorithm $\mathcal{B}$

1. Let m be the index of the layer which divides S in two parts, each of them with less than $|S|/2$ elements, that is, m is such that $|\bigcup_{i=1}^{m-1} L_i| < |S|/2$ and $|\bigcup_{i=m+1}^{h} L_i| < |S|/2$. Denote these halves of S by $L_\downarrow$ and $L_\uparrow$, respectively.
2. Perform a query about u to each $s \in L_m$:
   (a) if the outcome of one of these queries is hit, the search is over;
   (b) if the outcome of one of these queries is smaller, restart the search, restricted to $L_\downarrow$;
   (c) if the outcome to all these queries is no, restart the search, restricted to $L_\uparrow$.

Let us call the above strategy an *extended binary search*, and denote the algorithm which computes the respective decision tree by $\mathcal{B}(S, \prec)$. We clearly have $H_{\mathcal{B}}(n, m) \le w \lceil \lg h \rceil$; moreover, as the layering of S can be produced in time $O(m)$ and the building of the tree takes one step per element of S, we have $T_{\mathcal{B}}(n, m) = O(n + m)$.

We now turn our attention to another strategy: suppose we have $\{d_i\}_{i=1}^{k} \subseteq S$ such that $\{d_1\} = I(d_1) \subset I(d_2) \subset I(d_3) \subset \ldots \subset I(d_k) \subseteq S$. In this case we define the *segments* S_i $(0 \le i \le k)$ of $(S, \prec)$ by

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

We can then formulate the following algorithm which takes advantage of the above structure.

Algorithm $\mathcal{A}$

1. Perform a (usual) binary search of u through $\{d_i\}_{i=1}^{k}$; if u is found stop, otherwise let i be the minimum index such that $u \prec d_i$ (or k, if there is no such index).
2. Perform an extended binary search through the segment S_i.

We have

$$H_{\mathcal{A}}(n, m) \le \lceil \lg k \rceil + \max_{1 \le i \le k} H_{\mathcal{B}}(|S_i|, m) \le \lg n + w \lg \left(\max_{1 \le i \le k} |S_i| \right),$$

and this tree can be built in $T_{\mathcal{B}}(n, m) = O(n + m)$ steps.

In the following sections we introduce the *random graph order* model for partial orders and show that, in this model, we almost surely have

$$H_{\mathcal{A}}(n,m) = \lg n + O(\sqrt{\log n}\log\log n).$$

3.1 The Random Graph Model

The *random graph order* probability space, denoted $\mathcal{P}_{n,p}$, is the probability space of all partial orders $([n],\prec)$ obtained by independently choosing each pair of $\{(i,j)\in[n]^2 : i<j\}$ with probability p and taking the transitive closure of the resulting relation. We denote a generic partial order $([n],\prec)$ in $\mathcal{P}_{n,p}$ by $P_{n,p}$.

We say that $d\in[n]$ is *k-dominating* if $[d-k,d-1]\subseteq I(d)$ and call d a *dominating* element of $[n]$ if d is $(d-1)$-dominating (that is, if $x\prec d$ for all $1\le x<d$). The conditional probability that d should be k-dominating, given that d is $(k-1)$-dominating, is $1-(1-p)^k$, which leads us to

$$\mathbb{P}(d \text{ is } k\text{-dominating}) = (1-(1-p)^k)\mathbb{P}(d \text{ is } (k-1)\text{-dominating}).$$

By induction on k, we have

$$\mathbb{P}(d \text{ is } k\text{-dominating}) = \eta(k,p),$$

where $\eta(k,p)=\prod_{i=1}^{k}(1-(1-p)^i)$. We note that η is a strictly decreasing function of k and define $\eta(p)=\lim_{k\to\infty}\eta(k,p)$; then, for all $k\ge 1$, we have $\eta(p)<\eta(k,p)\le p$.

Lemma 1. *The probability that $d\in[n]$ is dominating is $\eta(d-1,p)$.*

As a consequence, we note that the expected number of dominating elements in $P_{n,p}$ is greater than $n\eta(n,p)$. By setting $\{d_i\}_{i=1}^{k}$ to be the dominating elements in $P_{n,p}$, we meet the necessary conditions for applying algorithm $\mathcal{A}$ on $P_{n,p}$.

In what follows, we will show that the decision tree built by $\mathcal{A}(P_{n,p})$ almost surely has "small height", by showing that both the size of each segment S_i and the size of each layer L_j are suitably small.

To show that the size of each segment is not too large we show that we have no large intervals of $[n]$ free of dominating elements in $P_{n,p}$; to show that the size of each layer is not too large, we use that we cannot have large antichains in $P_{n,p}$.

On the Size of the Segments. Let us consider the case in which d is not dominating. In this case, there must be a minimum b such that d is $(b-1)$-dominating but it is not b-dominating. If we call such b a *barrier* for (the domination of) d, then we have, for each $0<b<d-1$,

$$\begin{aligned}\mathbb{P}(b \text{ is a barrier for } d) &= \mathbb{P}(d-b\not\prec d \textbf{ and } d \text{ is } (b-1)\text{-dominating})\\ &= \mathbb{P}(d-b\not\prec d \mid d \text{ is } (b-1)\text{-dominating})\\ &\qquad\qquad\qquad \times \mathbb{P}(d \text{ is } (b-1)\text{-dominating})\\ &= (1-p)^b\eta(b-1,p).\end{aligned}$$

Thus, for any $0 < s < d-1$, we have that the probability that d is not dominating but d is s-dominating is

$$\begin{aligned}
\mathbb{P}\left(\bigvee_{b=s+1}^{d-2}\Big\{b \text{ is a barrier for } d\Big\}\right) &= \sum_{b=s+1}^{d-2}\mathbb{P}(b \text{ is a barrier for } d) \\
= \sum_{j=0}^{d-s-3}\mathbb{P}(j+s+1 \text{ is a barrier for } d) &= \sum_{j=0}^{d-s-3}(1-p)^{j+s+1}\eta(j+s,p) \\
\le \eta(s,p)(1-p)^{s+1}\sum_{j\ge 0}(1-p)^j &= \frac{\eta(s,p)}{p}(1-p)^{s+1} \le (1-p)^{s+1}.
\end{aligned}$$

Now, if $M \subseteq [n]$ and $d = \min\{|m-m'| \colon m,\, m' \in M\}$, then the events "$m$ is d-dominating" are mutually independent for all $m \in M$. For convenience, let D and D_d denote the set of dominating and d-dominating elements in $P_{n,p}$. We have

$$\begin{aligned}
\mathbb{P}(D\cap M = \emptyset) = \mathbb{P}(D\cap M=\emptyset \mid D_d\cap M=\emptyset)&\mathbb{P}(D_d\cap M=\emptyset) \\
&+ \mathbb{P}(D\cap M=\emptyset \text{ and } D_d\cap M\neq\emptyset) \\
\le \mathbb{P}\left(\bigwedge_{m\in M}\{m\notin D_d\}\right) &+ \mathbb{P}\left(\bigvee_{m\in M}\{m\notin D \text{ and } m\in D_d\}\right) \\
\le (1-\eta(d,p))^{|M|} &+ \sum_{m\in M}(1-p)^{d+1} \\
= (1-\eta(d,p))^{|M|} &+ |M|(1-p)^{d+1}.
\end{aligned} \tag{1}$$

Let us put

$$\beta(p) = \lg\frac{1}{1-p}, \qquad \gamma(p) = \lg\frac{1}{1-\eta(p)}, \qquad \alpha(p) = \frac{2}{\beta(p)\gamma(p)}.$$

Theorem 2. *If $g \ge \left\lceil \alpha(p)(\lg n\omega(n))^2\right\rceil$ and $x < n-g$, the probability that the set $[x, x+g]$ has no dominating element is at most $1/n\omega(n)$.*

Proof. If

$$d \ge \frac{\gamma(p)m + \lg m}{\beta(p)} - 1, \tag{2}$$

then $m(1-p)^{d+1} \le (1-\eta(p))^m$, so that if $|M| = m$ in (1) then the probability that there is no dominating element in M is $2(1-\eta(p))^m$. If

$$m \ge \frac{\lg 2n\omega(n)}{\gamma(p)}, \tag{3}$$

then the probability that there is no dominating elements in M is $\le (n\omega(n))^{-1}$.

We can then set $M = \{x + id\}_{i=0}^{m-1}$, with d and m satisfying (2) and (3) so that we have $M \subseteq [x, x+g]$ with

$$g \geq dm \geq \frac{(\lg 2n\omega(n))^2 + (\lg 2n\omega(n))\left(\lg\lg 2n\omega(n) - \left(\frac{1}{\beta(p)} + \lg\gamma(p)\right)\right)}{\beta(p)\gamma(p)}$$
$$\geq \frac{2(\lg n\omega(n))^2}{\beta(p)\gamma(p)} = \alpha(p)(\lg n\omega(n))^2.$$

□

Corollary 3. *The probability that $P_{n,p}$ has a segment of size $\geq \alpha(p)(\lg n\omega(n))^2$ is at most $1/\omega(n)$.*

Proof. Let $\{S_j\}_{j=1}^k$ be the segments of $P_{n,p}$. Then

$$\mathbb{P}\left(\bigvee_{j=1}^{k}\left\{|S_j| > \alpha(p)(\lg n\omega(n))^2\right\}\right) \leq \sum_{j=1}^{k}\mathbb{P}(|S_j| > \alpha(p)(\lg n\omega(n))^2)$$
$$\leq \sum_{j=1}^{k}\frac{1}{n\omega(n)} = \frac{k}{n\omega(n)} \leq \frac{1}{\omega(n)}.$$

□

On the Size of the Layers. Consider a layer L_i. Since L_i is an antichain in $P_{n,p}$, we can make direct use of the following result from Barak and Erdős [7].

Theorem 4 (Barak and Erdős [7]). *The probability that $P_{n,p}$ has an antichain of size larger than*

$$K_n = \sqrt{\frac{2\lg n}{\beta(p)} + \frac{1}{4}} + \frac{1}{2} \tag{4}$$

tends to zero as $n \to \infty$.

The Decision Tree is not Too Tall. We are now in position to prove the main result of this section.

Theorem 5. *The decision tree built by algorithm $\mathcal{A}(P_{n,p})$ has height almost surely bounded by $\lg n + O(\sqrt{\log n}\log\log n)$.*

Proof. Let H be the random variable in $\mathcal{P}_{n,p}$ whose value is the height of the decision tree built by the algorithm on the input $P_{n,p}$. As has been noted, we have $H \leq \lg n + w \lg \max_{1\leq i\leq h}|S_i|$, where w is the size of the greatest layer of $P_{n,p}$, the S_i are its segments, and h is its height.

Corollary 3 tells us that

$$\mathbb{P}\left(\max_{1\le i\le h}|S_i| > \alpha(p)(\lg n\omega(n))^2\right) < \frac{1}{\omega(n)},$$

while Theorem 4 gives

$$\mathbb{P}\left(w > \frac{1}{2} + \sqrt{\frac{2\lg n}{\beta(p)} + \frac{1}{4}}\right) < \frac{1}{\omega'(n)},$$

for some function $\omega' : \mathbb{N} \to \mathbb{R}$ satisfying $\lim_{n\to\infty}\omega'(n) = \infty$. Therefore

$$\begin{aligned}
&\mathbb{P}\left(\max_{1\le i\le h}|S_i| \le \alpha(p)(\lg n\omega(n))^2 \textbf{ and } w \le \frac{1}{2} + \sqrt{\frac{2\lg n}{\beta(p)} + \frac{1}{4}}\right)\\
&\quad = 1 - \mathbb{P}\left(\max_{1\le i\le h}|S_i| > \alpha(p)(\lg n\omega(n))^2 \textbf{ or } w > \frac{1}{2} + \sqrt{\frac{2\lg n}{\beta(p)} + \frac{1}{4}}\right)\\
&\quad < 1 - \left(\frac{1}{\omega(n)} + \frac{1}{\omega'(n)}\right) = 1 - o(1).
\end{aligned}$$

We conclude that

$$\lim_{n\to\infty}\mathbb{P}\left(H \le \lg n + \left(\frac{1}{2} + \sqrt{\frac{2\lg n}{\beta(p)} + \frac{1}{4}}\right)\lg(\alpha(p)(\lg n)^2)\right) = 1.$$

□

4 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 by U_n.

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 $([n], \prec)$ satisfying the following conditions:

– whenever $x \prec y$, for $x \in X_i$ and $y \in X_j$, we have $i < j$,
– whenever $x \in X_1$ and $y \in X_3$ we have $x \prec y$.

The partial orders in $\mathcal{A}(X_1, X_2, X_3)$ are said to be 3-*layered.*

Answering the question "what does a 'typical' partial order on $[n]$ look like?", in [8] Kleitman and Rothschild proved that, rather surprisingly, *almost all partially ordered sets are 3-layered.*

Theorem 6 (Kleitman and Rothschild [8]). *Suppose $\omega(n) \to \infty$ 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 $\big||X_2| - n/2\big| < \omega(n)$ and $\big||X_1| - n/4\big| < \omega(n)\sqrt{n}$.*

Theorem 6 makes the problem of searching in typical partial orders as posed in Section 1 rather uninteresting, since our search model makes it unavoidable to query each of the maximal elements of the given order. Theorem 6 tells us that almost all orders have $(1/4 + o(1))n$ such elements.

To make the problem more interesting, we now consider a variation 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; a strategy, accordingly, is redefined to be a decision tree as described previously, with the difference that each internal node has three children. In this section we shall prove that it is almost always possible, under the uniform model, to construct a ternary decision tree of height $O(\log n)$ in time $O(n^2 \log n)$.

Let us first make the following definition: given two disjoint sets X_1, X_2, the set of *2-layered orders* $\mathcal{A}(X_1, X_2)$ is the set of all partial orders $(X_1 \cup X_2, \prec)$ such that $\prec \subseteq X_1 \times X_2$.

Given a partition $\{X_1, X_2, X_3\}$ of $[n]$, there is a natural correspondence between $\mathcal{A}(X_1, X_2, X_3)$ and $\mathcal{A}(X_1, X_2) \times \mathcal{A}(X_2, X_3)$, so that we can devise a search strategy for a 3-layered order as a 'composition' of search strategies for 2-layered orders, by applying the latter on the suborders induced by $X_1 \cup X_2$ and $X_2 \cup X_3$.

Our strategy for searching for $u \in U$ in $P \in \mathcal{A}(X_1, X_2)$ is simple: we start by making the assumption $u \in X_1$, which we then verify by means of a series of queries to elements of X_2 in such a way that, at the end, we either have found u in X_1 or know that $u \notin X_1$. In the latter case, we restart the algorithm with the rôles of X_1 and X_2 exchanged, so that, after this is done, we either have found u in $X_1 \cup X_2$ or can conclude that $u \notin X_1 \cup X_2$. The algorithm to search for $u \in U$ in the layer X_1, say, starts by setting $S = X_1$ and proceeds in two phases.

Algorithm $\mathcal{C}$

```
querying phase: at each step,
    1. choose $x \in X_2$ such that $\big||S \cap I(x)| - |S - I(x)|\big|$ is minimal.
    2. query $x$ about $u$:
        (a) if the answer is hit, the search is over;
        (b) if the answer is smaller, replace $S$ with $|S \cap I(x)|$ and restart;
        (c) if the answer is no, replace $S$ with $|S - I(x)|$ and restart.
    This procedure is repeated until we reach the point where $S \cap I(x) = \emptyset$ or
    $S - I(x) = \emptyset$ for all $x \in X_2$, at which point we go to the next phase.
sweeping phase: for each $s \in S$:
    1. query $s$ about $u$: if the answer is hit, the search is over, otherwise, proceed.
    If the search is not over after this, we can conclude that $u \notin X_1$.
```

Let us call $s_i(X_1, X_2)$ the size of the set S at the beginning of the ith step of the querying phase, $s(X_1, X_2)$ the size of the set S at the beginning of the sweeping phase and $q(X_1, X_2)$ the number of queries made at the querying phase. It is clear that the height of the decision tree corresponding to the above procedure is bounded by $q(X_1, X_2) + s(X_1, X_2)$; moreover, we note that the choice of x at step i of the querying phase can be accomplished in $s_i(X_1, X_2)|X_2|$ steps.

From this and the above discussion, we can conclude that it is possible to construct a decision tree of height

$$H_{\mathcal{C}} \leq \sum_{(i,j)\in J} \big(q(X_i, X_j) + s(X_i, X_j)\big), \tag{5}$$

and this can be done in time

$$T_{\mathcal{C}} \leq \sum_{(i,j)\in J} \left(s(X_i, X_j) + |X_j| \sum_{k=1}^{q(X_i,X_j)} s_k(X_i, X_j) \right), \tag{6}$$

where $J = \{(1, 2), (3, 2), (2, 1)\}$.

At this point we define the *binomial probability model* $\mathcal{A}_{1/2}(X_1, X_2)$ for 2-layered orders, given by independently choosing each $(x_1, x_2) \in X_1 \times X_2$ with probability 1/2. We shall show that, in this model, we almost always have

1. $s_i(X_1, X_2) \leq (2/3)^i |X_1|$,
2. $q(X_1, X_2) \leq \log_{3/2} |X_1|$,
3. $s(X_1, X_2) < 40$,

which, along with (5), (6) and the sizes of X_1, X_2 and X_3 in Theorem 6 show that

$$H_{\mathcal{C}} \leq 3 \log_{3/2} n + 120 \qquad \text{and} \qquad T_{\mathcal{C}} \leq \frac{3}{2} n|X_2| + 120.$$

There is a natural correspondence between the product space $\mathcal{A}_{1/2}(X_1, X_2) \times \mathcal{A}_{1/2}(X_2, X_3)$ and the uniform probability space $\mathcal{A}(X_1, X_2, X_3)$. On the other hand, Theorem 6 and a standard argument give that if an event is almost certain in the space $\mathcal{A}(X_1, X_2, X_3) = \mathcal{A}_{1/2}(X_1, X_2) \times \mathcal{A}_{1/2}(X_2, X_3)$ for *any* fixed partition (X_1, X_2, X_3) as in Theorem 6, then this event must also be an almost certain event in $\mathcal{U}_n$ (details are given in the final version [9] of this extended abstract).

Therefore, in the remaining of this section, we concentrate in proving the relevant results for $\mathcal{A}_{1/2}(X_1, X_2) \times \mathcal{A}_{1/2}(X_2, X_3)$, where the X_i are as in the Kleitman–Rothschild theorem.

In what follows, we assume $S \subseteq X_1$, $x \in X_2$ and let $d_S(x) = |I(x) \cap S|$ be the number of elements in S smaller than x. The reader may notice that it is easy to prove analogous versions of the proposition and corollary below with X_1 and X_2 interchanged. Moreover, both, proposition and corollary, obviously remain true if we consider X_3 instead of X_1.

Proposition 7. *Almost surely, for a fixed $S \subseteq X_1$ with $|S| \geq 40$, the probability that there is no vertex $x \in X_2$ with $|S|/3 \leq d_S(x) \leq 2|S|/3$ is*

$$\leq \left(2\exp\left\{-|S|/40\right\}\right)^{|X_2|}.$$

Proof. Fix $S \subseteq X_1$ and $x \in X_2$. Then, by Chernoff's inequality, we have

$$\mathbb{P}\left(|d_S(x) - \mu| > |S|/6\right) < 2\exp\left\{-\mu/20\right\},$$

where $\mu = |S|/2$ is the expected value of $d_S(x)$. Thus, we have

$$\mathbb{P}\big(\nexists x \in X_2 \colon |S|/3 \leq d_S(x) \leq 2|S|/3\big) \leq \left(2\exp\left\{-|S|/40\right\}\right)^{|X_2|},$$

as required. □

Corollary 8. *For X_1, X_2 as above, and q, s_i, s be as defined in the preceding discussion, the following almost surely holds:*

1. $s(X_1, X_2) \leq 40$,
2. $s_i(X_1, X_2) \leq (2/3)^i |X_1|$, *and so,*
3. $q(X_1, X_2) \leq \log_{3/2} |X_1|$.

Proof. To prove (1) we observe that the probability of $s(S, X_2) > 40$, for some $S \subset X_1$ with $|S| > 40$, is given by

$$\begin{aligned}
&\mathbb{P}\left(\exists S \subseteq X_1 \, \forall x \in X_2 \text{ we have } \big|d_S(x) - |S|/2\big| > |S|/6\right) \\
&\quad\leq \sum_{S \subseteq X_1,\ |S| \geq 40} \mathbb{P}\left(\forall x \in X_2 \text{ we have } \big|d_S(x) - |S|/2\big| > |S|/6\right) \\
&\quad\leq \sum_{s=40}^{|X_1|} \binom{|X_1|}{s} \exp\left\{-\left(s/40 - \ln 2\right)|X_2|\right\} \\
&\quad\leq \sum_{s=40}^{|X_1|} \exp\left\{-\left(s/40 - \ln 2\right)|X_2| + s \ln |X_1|\right\} = o(1).
\end{aligned}$$

To prove (2) and (3), it suffices to notice the following. The probability that, at some point in step 1 of the querying phase, we cannot choose $x \in X_2$ such that $|S|/3 \leq |S \cap I(x)| \leq 2|S|/3$ is at most

$$(\log_{3/2} |X_1|)\mathbb{P}\left(\forall x \colon \big|d_S(x) - |S|/2\big| > |S|/6\right) \leq n \left(\frac{2}{e}\right)^{|X_2|} = o(1).$$

Thus in both cases (b) and (c) of step 2 of algorithm $\mathcal{C}$, we have reduced the search space by a factor $\leq 2/3$. Therefore we have $q(X_1, X_2) \leq \log_{3/2} |X_1|$. □

Theorem 9. *Almost every U_n admits a ternary search tree of height $O(\log n)$ which can be constructed in time $O(n^2)$.*

5 Concluding Remarks

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 [10] and [11] are crucial. We shall come back to this in [9].

The arguments in the proofs of Corollary 3 and preceding lemmas and theorems are adapted rewritings of arguments found in [12] (Lemma 2.3 and Theorems 2.10 and 2.11). The results in that paper, on the structure of random graph orders, were what first suggested to the authors the idea of the algorithm.

In the present work we consider only the case in which p does not depend on n. It should be noted, however, that the results in [12] imply that ours remain valid even in the case in which $p = p(n)$ is a decreasing function of n, as long as $p \lg n \to \infty$.

It is worth noting that Theorem 4 is part of a deeper investigation on the width of random graph orders found in [7], where a much stronger result is proven, namely, that the width of a random graph order is an almost determined random variable whose value, rather surprisingly, is almost surely $\lfloor K_n \rfloor$ or $\lceil K_n \rceil$, where K_n is given in (4).

As mentioned in Section 2, Linial and Saks [2] consider a different although related problem where the set U is assumed to be totally ordered, the given relation $(S, \prec)$ is a partial order compatible with this total order and where each query about $u \in U$ to $s \in S$ is made with respect to the total order induced on S and not with respect to the given relation $\prec$ 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 underlying total order in S, the outcome smaller leaves all elements in $S - F(s)$ as valid candidates.

While not presenting explicitly an algorithm to compute an optimal decision tree for the problem, it is a consequence of the results in [2] and [6] that the height H of an optimal decision tree for their problem satisfies

$$\lg \iota(S, \prec) \leq H \leq k \lg \iota(S, \prec), \tag{7}$$

where $\iota(S, \prec)$ is the number of ideals in $(S, \prec)$ and $k = (2 - \lg(1 + \lg 5))^{-1} \approx 3.73$.

We note that this fact alone can lead us to similar results to those given in Section 3 when we consider *their* problem in the random graph order model. Let us briefly discuss this. Redefine an element of S to be *dominating* if it is comparable to every other element in $(S, \prec)$, and a *segment* to be an interval free of dominating elements.

A possible search strategy, then, would be isolating one segment by means of a binary search restricted to the dominant elements of the order and then searching through this segment.

An ideal in 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 if w is the width of the order, then

$$\iota(R, \prec) \leq \sum_{i=1}^{w} \binom{|R|}{i} \leq |R|^w.$$

Together with the bounds in (7) and Theorem 4, this allows us to conclude that there is a decision tree for R of height at most

$$H \leq k \lg \iota(R, \prec) \leq kw \lg |R|.$$

Therefore we can conclude that, for any partial order, there is a decision tree of height $H \leq \lg n + w \lg s$, where s is the size of the largest segment in the order.

Now, the argument leading to Theorem 2 proves a result for dominating elements as defined now (with another value for $\alpha(p)$, of course, say $\alpha'(p)$), which, together with Theorem 4 gives us

$$\lim_{n\to\infty} \mathbb{P}\left(H \leq \lg n + k\left(\frac{1}{2} + \sqrt{\frac{2\lg n}{\beta(p)} + \frac{1}{4}}\right)\lg(\alpha'(p)(\lg n)^2)\right) = 1.$$

References

1. Laber, E.S., Nogueira, L.T.: Binary searching in posets. Submitted for publication (2001)
2. Linial, N., Saks, M.: Searching ordered structures. Journal of Algorithms **6** (1985) 86–103
3. Ben-Asher, Y., Farchi, E., Newman, I.: Optimal search in trees. SIAM Journal on Computing **28** (1999) 2090–2102
4. Ben-Asher, Y., Farchi, E.: The cost of searching in general trees versus complete binary trees. Technical Report 31905, Technical Report Research Center (1997)
5. Lipman, M.J., Abrahams, J.: Minimum average cost testing for partially order. IEEE Transactions on Information Theory **41** (1995) 287–291
6. Linial, N., Saks, M.: Every poset has a central element. Journal of Combinatorial Theory, Series A **40** (1985) 195–210
7. Barak, A.B., Erdős, P.: On the maximal number of strongly independent vertices in a random acyclic directed graph. SIAM Journal on Algebraic and Discrete Methods **5** (1984) 508–514
8. Kleitman, D.J., Rothschild, B.L.: Asymptotic enumeration of partial orders on a finite set. Trans. Amer. Math. Soc. **205** (1975) 205–220
9. Carmo, R., Donadelli, J., Laber, E., Kohayakawa, Y.: Searching in random partially ordered sets. In preparation (2002)
10. Prömel, H.J., Steger, A., Taraz, A.: Counting partial orders with a fixed number of comparable pairs. Combin. Probab. Comput. **10** (2001) 159–177
11. Prömel, H.J., Steger, A., Taraz, A.: Phase transitions in the evolution of partial orders. J. Combin. Theory Ser. A **94** (2001) 230–275
12. Bollobás, B., Brightwell, G.: The structure of random graph orders. SIAM Journal on Discrete Mathematics **10** (1997) 318–335