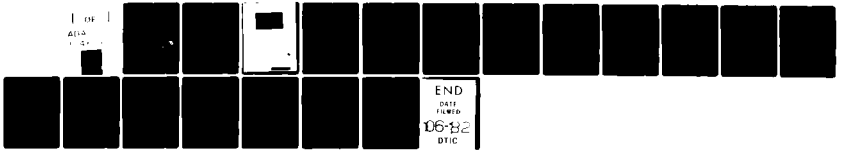
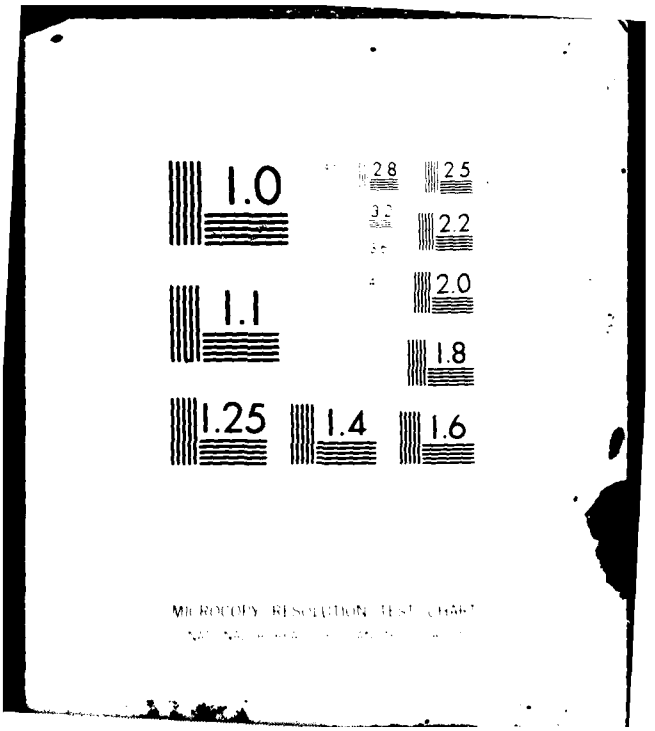
AD-A114 611 HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB F/G 12/1
PARALLEL INTERPOLATION SEARCH.(U)
MAR 82 J H REIF N00014-80-C-0674
UNCLASSIFIED TR-32-81 NL



END
DATE FILMED
06-82
DTIC


1.0
2.8
2.5
3.2
2.2
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
	AD-A114 601	

4. TITLE (and Subtitle)
Parallel Interpolation Search

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER
TR-32-81

7. AUTHOR(s)
John H. Reif

8. CONTRACT OR GRANT NUMBER(s)
N00014-80-C-0674

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Harvard University
Cambridge, MA 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

12. REPORT DATE
March, 1982

13. NUMBER OF PAGES
14

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
same as above

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
parallel search, interpolation algorithm, expected time.

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)
see reverse side

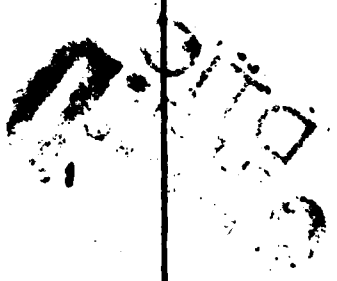
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

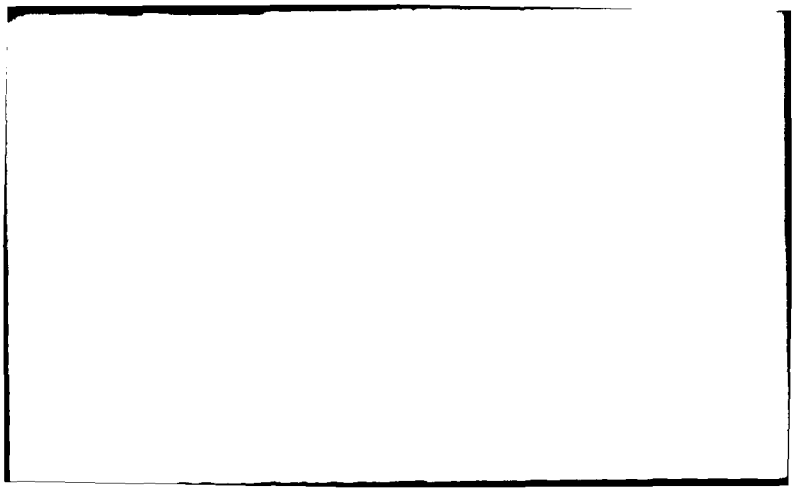
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

Abstract

This paper concerns the problem of searching, with p parallel processors, for a given key in a random ordered table of size n. We propose a parallel interpolation algorithm which we show has expected time cost $\leq \log(1 + \log(n)/\log(p)) + O(1)$ and we prove this algorithm has optimal expected time cost within a constant additive term.





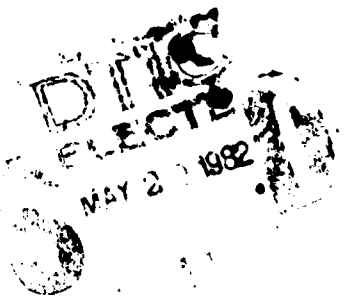
(2)

PARALLEL INTERPOLATION SEARCH

John H. Reif

TR-07-82

March 1982



A

PARALLEL INTERPOLATION SEARCH

by John H. Reif*

Aiken Computation Lab.
Division of Applied Science
Harvard University

March 1982

Abstract

This paper concerns the problem of searching, with p parallel processors, for a given key in a random ordered table of size n. We propose a parallel interpolation algorithm which we show has expected time cost $\leq \log(1 + \log(n)/\log(p)) + O(1)$ and we prove this algorithm has optimal expected time cost within a constant additive term.

*This work was supported by National Science Foundation Grant NSF-MCS79-21024 and the Office of Naval Research Contract N00014-80-C-0647.

1. INTRODUCTION

1.1 The Parallel Search Problem for a Random Ordered Table

We assume a *random ordered table*

$$(L, X_1, \ldots, X_n, H)$$

where $L = X_0$ and $H = X_{n+1}$ are given reals, $L < H$, and $X_1 < \ldots < X_n$ are constructed by sorting n distinct random reals chosen from $\{x \mid L < x < H\}$. The table has *size* n. Given a *search key* y, $L < y < H$, we wish to determine the index k^* such that $X_{k^*} \leq y < X_{k^*+1}$.

We assume a parallel machine model with $p \geq 1$ synchronized processors which in a single step may simultaneous read p distinct table keys $X_{k_1}, \ldots, X_{k_p}$ at indices $k_1, \ldots, k_p$ determined by the algorithm. The algorithm must then utilize these values to determine the indices of the table keys to be read in the next step. (Note that as in the previous literature on search algorithms, for example [Yao and Yao, 76], [Perl and Itai, 78], and [Gonnet, Rogers and George, 80], we do not take into account the cost to calculate these indices $k_1, \ldots, k_p$.)

The *worst case time* for a parallel search algorithm is the maximum of the number of steps required for any key y searched in any ordered table of size n. The *expected time* is the maximum, for any given search key y, of the average number of steps for searching y in a random ordered table of size n.

1.2 Binary Search

[Knuth, 73] shows that sequential binary search has time bound $\lfloor \log(n) \rfloor$ in the worst case and $\log(n)$ in the average, and that this is optimal for the sequential comparison tree model. [Gal and Miranker,

67] and [Snir, 87] have shown that a parallel binary search has worst case time $\leq \log(n)/\log(p)$ and we can also show this to be the average time for parallel binary search. [Snir, 82] has also shown that any parallel search algorithm must have worst case time $\geq \log(n)/\log(p)$ so parallel binary search gives optimal worst case time. However, we shall see that parallel binary search is extremely nonoptimal with respect to expected time.

1.3 Interpolation Search

[Peterson, 57] is the first published account of the use of repeated interpolations to choose indices at expected locations of the search key x. The expected time complexity of sequential interpolation search was posed as an open problem in [Knuth, 73]; subsequently it was independently shown to be $\log\log(n) + O(1)$ by [Yao and Yao, 76], [Perl and Itai, 78] and [Gonnet, Rogers, and George, 80]. [Yao and Yao, 76] proved that interpolation search has optimal expected time for any sequential search algorithm, within a constant additive term. Also, they found that a constant number of processors do not speed up interpolative search by more than a constant additive factor. They pose as an open problem the expected time complexity of searching with p processors, when p is a function of n.

1.4 Results of this Paper

In this paper we propose two algorithms for parallel search:

(a) *Parallel Pseudo Interpolative Search* is a generalization of a sequential pseudo interpolation search of [Perl and Reingold, 77]. We show in Section 2 that this parallel search algorithm has expected time

$\leq C(\varepsilon)\log(1+\log(n)/(4\log(\varepsilon p))) + O(1)$ where $\varepsilon > 0$ is a parameter of the algorithm which may be set arbitrarily and $C(\varepsilon) \geq 1$ is a constant that approaches 1 exponentially fast as $\varepsilon \to 0$. For example $C(1) \leq 2.03$.

(b) *Parallel Interpolation Search* is the natural generalization of the sequential interpol ation search. We show this parallel search algorithm has expected time $\leq \log(1+\log(n)/\log(p)) + O(1)$.

In Section 4 we show that any parallel search algorithm requires time $\geq \log(1+\log(n)/\log(p)) - c$, where $c \geq 0$ is a constant. Thus our parallel interpolation search algorithm has optimal expected time.

Section 5 concludes this paper.

2. PARALLEL PSEUDO INTERPOLATION SEARCH ($PIS_{\varepsilon,p}$)

This section describes a parallel algorithm $PIS_{\varepsilon,p}$ for searching by using repeated phases of a single interpolation followed by a series of parallel probes at equally spaced intervals, shifted linearly from the interpolation. Our algorithm is a generalization of the sequential pseudo interpolation search algorithm of [Perl and Reingold, 77] where here we utilize p processors and introduce a parameter $\varepsilon > 0$ which may be fixed to improve the performance of the algorithm ([Perl and Reingold, 77] only describe the case where $p = 1$ and $\varepsilon = 1$).

2.1 Algorithm $PIS_{\varepsilon,p}$

Input random ordered table $(L = X_0, X_1, \ldots, X_n, H = X_{n+1})$ and search key y, $L < y < H$.

Output index k^* where $X_{k^*} \leq y < X_{k^*+1}$.

Note that k^* is a binomial random variable whose distribution function has parameters α, n where $\alpha = (y - L)/(H - L)$. Thus $\mu = \alpha n$ is the expected value of k^*. Our immediate goal will be to determine k^* within bounds of distance $\delta = \sqrt{n}/(\varepsilon p)$. These bounds will be further reduced by recursive calls to the algorithm. Define indices $k_i = \lceil \mu + i\delta \rceil$ for all integers i, $\ell_0 \leq i \leq h_0$, where $\ell_0 = \lceil 1 - \mu/\delta \rceil$ and $h_0 = n - \lfloor \mu/\delta \rfloor$.

We will repeatedly execute the following *loop*:

Intially let $i_0 \leftarrow 0$. In parallel read values X_{k_i} for each i, $\ell \leq i \leq h$, where $\ell = \max(\ell_0, i_0 - \lfloor p/2 \rfloor + 1)$ and $h = \min(h_0, i_0 + \lceil p/2 \rceil)$. If $y < X_{k_\ell}$ than repeat the above step with $i_0 \leftarrow i_0 - p$. If $y > X_{k_h}$ then repeat the above step with $i_0 \leftarrow i_0 + p$. Otherwise there exists i, $i+1$ where $\ell \leq i$, $i+1 \leq h$ and $X_{k_i} \leq y < X_{k_{i+1}}$. If $\delta = 1$ then halt with output

$k^* = k_i$ and otherwise recursively execute $PIS_{\varepsilon,p}$ to search for y in table $(X_{k_i}, X_{k_i+1}, \ldots, X_{k_{i+1}-1}, X_{k_{i+1}})$.

2.2 Probabilistic Analysis of $PIS_{\varepsilon,p}$

In the following we fix the number of processors $p \geq 2$. Let S_ε be the random variable giving the number of steps the *loop* of algorithm $PIS_{\varepsilon,p}$ is executed until there are determined i, $i+1$ such that $X_{k_i} \leq y < X_{k_i+1}$. Let $C(\varepsilon) = \text{mean}(S_\varepsilon)$.

LEMMA 1.

$$C(\varepsilon) \leq 1 + \sum_{s=1}^{n} \frac{\varepsilon p}{s\sqrt{2\pi}} e^{-2(s/\varepsilon p)^2} .$$

Proof. By definition,

$$C(\varepsilon) = \sum_{s \geq 1} s \cdot \text{Prob}\{S_\varepsilon = s\}$$

$$= \sum_{s \geq 1} \text{Prob}\{S_\varepsilon \geq s\} .$$

Since the loop is always executed at least once, $S_\varepsilon \geq 1$. We have already noted that k'' has a binomial distribution with parameters α, p and thus variance $\sigma^2 = \alpha(1-\alpha)n$. We can approximate the distribution of k^* by a normal distribution (see [Feller, 68]), giving for $s \geq 2$

$$\text{Prob}\{S_\varepsilon \geq s\} \leq 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-\frac{1}{2}z^2} dz \leq \frac{1}{4\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$$

where

$$u = (s-1)\delta/\sigma$$

$$= (s-1)/(\varepsilon p\sqrt{\alpha(1-\alpha)})$$

$$\leq 2(s-1)/(\varepsilon p) \quad . \qquad \square$$

Let $\bar{T}_{\varepsilon,p}(n)$ be the expected number of steps executed by algorithm $PIS_{\varepsilon,p}$ for a table of size n.

THEOREM 1. $\bar{T}_{\varepsilon,p}(n) \leq C(\varepsilon)\log(1 + \log/(4\ \log(\varepsilon p))) + O(1)$ *for* $\varepsilon p > 2$.

Proof. Clearly $\bar{T}_{\varepsilon,p}(p) = 1$. Since each recursive call reduces a table of size n to a subtable of size $\delta = \sqrt{n}/(\varepsilon p)$ we have

$$\bar{T}_{\varepsilon,p}(n) \leq \bar{T}_{\varepsilon,p}(\lceil\sqrt{n}/(\varepsilon p)\rceil) + C(\varepsilon)$$

$$\leq \bar{T}_{\varepsilon,p}(\lceil(n^{1/2}\varepsilon p)^{2^{-t}}(\varepsilon p)^{-2}\rceil) + tC(\varepsilon)$$

$$= \bar{T}_{\varepsilon,p}(p) + tC(\varepsilon)$$

for

$$t \leq \log(1 + \log(n)/(4\ \log(\varepsilon p))) + O(1) \quad . \qquad \square$$

3. PARALLEL INTERPOLATION SEARCH (IS_p)

This Section presents a parallel search algorithm which seems a natural generization of the sequential interpolation search of [Peterson, 57] to p parallel processors.

3.1 Algorithm IS_p

Input random ordered table $(L = X_0, X_1, \ldots, X_n, H = X_{n+1})$ and search key y, $L < y < H$.

Output index k^* where $X_{k^*} \leqslant y < X_{k+1}$.

Intially assign $\ell \leftarrow 0$ and assign $h \leftarrow n + 1$.

Repeat forever the following *loop*:

We can assume that we have already read the values of X_ℓ and X_h and must search for y in the random ordered table $(X_\ell, X_{\ell+1}, \ldots, X_{h-1}, X_h)$. Assign $n' \leftarrow h - \ell - 1$. If $p \geqslant n'$ then all of $X_{\ell+1}, \ldots, X_{h-1}$ can be read in a single step so we can halt and output k^*. Otherwise, assign $\alpha \leftarrow (y - X_\ell)/(X_h - X_\ell)$. Note that $k^* - h$ has a binomial distribution with parameters α and n'. Let IB be the functional inverse of this binomial distribution function (i.e., $z = \text{Prob}\{k^* - \ell \leqslant IB(z)\}$ for $0 \leqslant z \leqslant 1$). Assign the indices $k_i \leftarrow \lceil \ell + IB(i/(p+1)) \rceil$ for $i = 1, \ldots, p$. Also let $k_0 \leftarrow \ell$ and $k_{p+1} \leftarrow h$. In a single parallel step read X_{k_i} for $i = 1, \ldots, p$. Assign d to be the maximum integer such that $X_{k_d} \leqslant y < X_{k_i}$ for all i, $d < i \leqslant p + 1$. Finally reassign $\ell \leftarrow k_d$, $h \leftarrow k_{d+1}$ and repeat the above loop.

3.2 Probabilistic Analysis of Algorithm IS_p

For each $t = 1,2,\ldots$ let $\ell(t)$, $h(t)$ be the values of ℓ, h respectively on entering the t-th iteration of the *loop* of IS_p, and let $n'(t)$, $\alpha(t)$, $IB^{(t)}$, $k_i(t)$, $d(t)$ be the values of the corresponding variables as defined in the t-th iteration of the *loop*. For notational simplicity let $k(t) = k_{d(t)}(t)$ and let $k(0) = 0$. Let $\Delta(t) = |k(t) - k(t-1)|$ and $\Delta_i(t) = |k_i(t) - k(t-1)|$ for $i = 1,\ldots,p$. By definition we have for $i = 1,\ldots,p$

PROPOSITION 1.

$$\Delta_i(t) = \begin{cases} B^{(t)}\left(\frac{i}{p+1}\right) & \text{for} \quad y > X_{k(t-1)} \\ n'(t) - B^{(t)}\left(\frac{i}{p+1}\right) & \text{otherwise.} \end{cases}$$

The *history sequence* previous to the t-th iteration is

$$\sigma(t) = ((X_{\ell(1)}, X_{h(1)}, \ell(1), h(1)), \ldots, (X_{\ell(t)}, X_{h(t)}, \ell(t), h(t)))\ .$$

For each $i = 1,\ldots,p$ let $\sigma_i(t)$ be the event $\sigma(t)$ and $d(t) \in \{i-1, i\}$. Again by definition it follows that for $i = 1,\ldots,p$

PROPOSITION 2. $\quad k_i(t) = E(k^* | \sigma_i(t))$.

Since each of the conditional events $(\sigma_i(t) | \sigma(t))$ are equally likely,

PROPOSITION 3.

$$\Delta(t) = \frac{1}{p} \sum_{i=1}^{p} \Delta_i(t) \quad .$$

From the above Propositions it follows that the expected error just before the t-th iteration is

PROPOSITION 4. $\Delta(t) = \left|E(k^* - k(t-1) \,|\, \sigma(t))\right|$.

We can bound $\Delta(t)$ on consecution iterations as follows:

LEMMA 2. $E((\Delta(t+1))^2 \,|\, \sigma(t)) \leqslant \Delta(t)/p$.

Proof.

$$E((\Delta(t+1))^2 \,|\, \sigma(t)) = E((E(k^*-k(t) \,|\, \sigma(t+1)))^2 \,|\, \sigma(t)) \quad \text{by Proposition 4}$$

$$\leqslant E((E(k^*-k(t))^2 \,|\, \sigma(t+1) \,|\, \sigma(t))$$

$$= E((k^*-k(t))^2 \,|\, \sigma(t))$$

$$\leqslant \frac{1}{p^2} \sum_{i=1}^{p} E((k^*-k(t))^2 \,|\, \sigma_i(t)) \quad \text{by Proposition 3}$$

$$\leqslant \frac{1}{p^2} \sum_{i=1}^{p} \frac{1}{n'(t)} IB^{(t)}\left(\frac{i}{p+1}\right)\left[n'(t) - IB^{(t)}\left(\frac{i}{p+1}\right)\right]$$

by computing the second moments of p clipped binomial distributions

$$\leqslant \frac{1}{p^2} \sum_{i=1}^{p} \min\left(IB^{(t)}\left(\frac{i}{p+1}\right),\ n'(t) - IB^{(t)}\left(\frac{i}{p+1}\right)\right)$$

$$\leqslant \frac{1}{p^2} \sum_{i=1}^{p} \Delta_i(t) \quad \text{by Proposition 1}$$

$$= \Delta(t)/p \quad \text{by Proposition 3.}\square$$

Let $\bar{T}_p(n)$ be the expected time of Algorithm IP_p.

THEOREM 2. $\bar{T}_p(n) \leqslant \log(1 + \log(n)/\log(p)) + O(1)$.

Proof. Define $\Gamma(t)$ to be a continuous function which is a linear interpolation of $2^t \log(\Delta(t)) - t \log(p)$ at distinct points $t = 0,1,\ldots$ where $\Delta(0) = n$. Then $\Gamma(t)$ is a supermartingale with respect to history sequence $\sigma(t)$ since by Lemma 2, $E(\Gamma(t+1)|\sigma(t)) \leq \Gamma(t)$. Let the *stopping time* T be the minimum $t \geq 0$ such that $\Gamma(t) \leq (2^t - t)\log(p)$, so $\Delta(\lceil T \rceil) \leq p$. Then by the Optimal Stopping Theorem (see [Karlin and Taylor, 75]),

$$E(\Gamma(T)) \leq E(\Gamma(0)) = \log(n) .$$

Thus

$$\tilde{T}_p(n) = E(T) \leq \log(1 + \log(n)/\log(p)) + 0(1) . \qquad \square$$

4. PARALLEL INTERPOLATION SEARCH IS OPTIMAL

This section shows that the parallel interpolation search algorithm described in Section 3 is optimal within a constant additive term. We actually prove a slightly stronger theorem for which our optimality result follows as a corollary.

Given a random ordered table $(L,X_1,\ldots,X_n,H)$ and a search key y, $L < y < H$, let the *expected distance* of this search problem be $\lambda = \lceil \alpha n \rceil$ where $\alpha = (y - L)/(H - L)$. For integers $p \geq 1$ and $\lambda \geq 1$ let $R_p(\lambda)$ be the minimum expected time cost for any parallel algorithm with p processors to solve all search problems with expected distance λ. The following theorem is a direct generalization of a lower bound proof due to [Yao and Yao, 76] for sequential search of random ordered tables.

THEOREM 3. $R_p(\lambda) \geq f_p(\lambda) - c + 1/f_p(\lambda)$ *where* $c \geq 0$ *is a constant and* $f_p(\lambda) = \log(1 + \log(\lambda)/\log(p))$.

Proof by induction. Suppose the theorem holds for all $\lambda' < \lambda$. Let $(L,X_1,\ldots,X_n,H)$ be a random ordered table with search key y and expected distance λ. Fix a parallel search algorithm A_p with p processors. Let $k_1,\ldots,k_p$ be the choice by A_p of key indices to be read on the first step. Then from these indices we can show there exists a set $J \subseteq [L,H]$ such that if HIT is the event $X_{k_i} \in J$ for some i,

(1) $\text{Prob}(\text{HIT}) \leq \lambda^{-2\varepsilon}$, and

(2) not HIT implies that on the second step of A_p the resulting search problem has expected distance $\geq \lambda'$, where

$$\lambda' = \frac{\lambda^{1/2-\varepsilon}}{p}$$

and

$$\varepsilon = (\ln(\lambda)/(2\ln(p)))^{-1/2}.$$

Thus

$$R_p(\lambda) \geq 1 + \text{Prob(not HIT)} \cdot R_p(\lambda')$$

$$\geq 1 + (1 - \lambda^{-2\varepsilon})(f_p(\lambda') - c + 1/f_p(\lambda'))$$

$$\geq f_p(\lambda) - c + 1/f_p(\lambda) \quad . \qquad \square$$

For any table size n, $R_p(\lambda)$ is maximum when the search key is chosen so $\lambda = \lceil n/2 \rceil$. By Theorem 3, we can always find a constant $c \geq 0$ such that

COROLLARY. *Any parallel search algorithm* A_p *with* p *processors has expected time*

$$\geq \log(1 + \log(n)/\log(p)) - c \ .$$

5. CONCLUSION

We have determined, within an additive constant, the average case complexity of parallel searching an ordered random table with keys independently chosen from a uniform distribution. Our results can be extended to nonuniform distributions which have invertible cumulative distribution functions.

Of the two algorithms we describe in this paper the parallel pseudo interpolation algorithm of Section 2 may be more practical since the interpolations require only calculation of the mean of a binomial whereas the parallel interpolation algorithm of Section 3 requires calculating the inverse of the cumulative distribution function of a binomial.

References

Doob, J.L., *Stochastic Processes*, Wiley, New York, 1967.

Feller, W., *An Introduction to Probability Theory and Its Applications*, Vol. I, Wiley, New York, third ed., 1968.

Gal, S., and W. Miranker, "Optimal Sequential and Parallel Search for Finding a Root," *J. of Comb. Th.* (A)23, 1977, 1-14. *IBM J. Res. Devel.* 13, 1967, 297-301.

Ghosh, S.P., and Senko, M.E. "File Organization: On the Selection of Random Access Index Points for Sequential Files," *JACM* 16 (1969), 569-579.

Gonnet, G.H., "Interpolation and Interpolation-Hash Searching," Ph.D. Thesis, Waterloo: University of Waterloo, 1977.

Gaston H. Gonnett, Lawrence D. Rogers, and J. Alan George, "An Algorithmic and Complexity Analysis of Interpolation Search," *Acta Informatica* 13, 39-52 (1980).

Karlin, S., and Taylor, H.M., *A First Course in Stochastic Processes*, Academic Press, New York, second ed., 1975.

Knuth, D.E., *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, Mass., 1973.

Kruijer, H.S.M., "The Interpolated File Search Method," *Informatie* 16, 612-615 (1974).

Perl, Y., and Reingold, E.M., "Understanding and Complexity of Interpolation Search," *Infrm. Proc. Letters*, 6 (1977), 219-222.

Perl, Y., Itai, A., Avni, H., "Interpolation Search--A Log Log N Search," *Comm. ACM* 21, 550-557 (1978).

Peterson, W.W., "Addressing for Random-Access Storage," *IBM Journal of Research and Development* 1, 130-146 (1957).

Snir, M., "On Parallel Search," Courant Institute, New York University, Technical Report, 1982.

Valiant, Leslie G., "Parallelism in Comparison Problems," *SIAM J. Comput.* Vol. 4, No. 3, September 1975.

Yao, A.C., and F.F. Yao, "The Complexity of Searching an Ordered Random Table," Proc. of the 17th Annual Symposium on Foundations of Computer Science (1976), 173-177.


ATE
LMED