INFORMATION AND COMPUTATION 81, 364–379 (1989)

Parallel Processing Can Be Harmful: The Unusual Behavior of Interpolation Search

DAN E. WILLARD*

State University of New York at Albany and Consultant to Bell Communications Research

AND

JOHN H. REIF†

Duke University

Several articles have noted the usefulness of a retrieval algorithm called sequential interpolation search, and Yao and Yao have proven a lower bound $\log\log N - O(1)$, showing this algorithm is actually optimal up to an additive constant on unindexed files of size N generated by the uniform probability distribution. We generalize the latter to show $\log\log N - \log\log P - O(1)$ lower bounds the complexity of any retrieval algorithm with P parallel processors for searching an unindexed file of size N. This result is surprising because we also show how to obtain an upper bound that matches the lower bound up to an additive constant with a procedure that actually uses *no parallel processing* outside its last iteration (at which time our proposal turns on P processors in parallel). Our first theorem therefore states that *parallel processing before the literally last iteration* in the search of an unindexed ordered file has *nearly no usefulness*. Two further surprising facts are that the preceding result holds even when communication between the parallel processing units involves *no delay* and that the parallel algorithms are actually *inherently slower* than their sequential counterparts when each invocation of the SIMD machine invokes a communication step with *any type* of nonzero *delay*. The presentation in the first two chapters of this paper is quite informal, so that the reader can quickly grasp the underlying intuition.

1. INTRODUCTION

Searching an ordered file is a very common operation in data processing and so should be accomplished as rapidly as possible. Given N records, stored (ideally) in successive memory locations in the order of their numeric keys $Y_1 < Y_2 \cdots < Y_N$, one often wishes to find a particular record

* Supported partially by NSF Grant IRI 87-03430.
† Supported partially by NSF Grant MCS79-21024 and ONR Contract N00014-80-C-0647.

0890-5401/89 $3.00

whose key equals y. For simplicity, the literature on interpolation search (Gonnet, 1977; Gonnet, Rogers, and George, 1980; Perl and Reingold, 1977; Perl, Itai, and Avni, 1978; Peterson, 1957; Yao and Yao, 1976) generally assumes these N records were generated by the uniform distribution over the (0, 1) interval and that this file is initially padded with the two pseudo-key boundary values of $Y_0 = 0$ and $Y_{N+1} = 1$. Each iteration i of interpolation search will search a subfile of the form $F_i = \{Y_{L_i} < Y_{L_i+1} < \cdots < Y_{R_i}\}$, where the contents of only the boundary keys Y_{L_i} and Y_{R_i} are known at the start of the ith iteration (for instance, the first iteration has $L_1 = 0$, $R_1 = N+1$, $Y_{L_1} = 0$, and $Y_{R_1} = 1$). Interpolation search's ith iteration will generate the cut address C_i^* below and check to see whether $Y_{C_i^*} = y$,

$$C_i^* = L_i + \left\lceil \frac{(R_i - L_i - 1)(y - Y_{L_i})}{Y_{R_i} - Y_{L_i}} \right\rceil. \tag{1.1}$$

If $Y_{C_i^*} > y$ then the next iteration of interpolation search will search the subfile $F_{i+1} = \{Y_{L_i} < Y_{L_i+1} < \cdots Y_{C_i^*}\}$, and it will search the subfile $F_{i+1} = \{Y_{C_i^*} < Y_{C_i^*+1} < \cdots Y_{R_i}\}$ when $Y_{C_i^*} < y$. This process will continue until interpolation search either finds a record storing the key y or reduces the search problem to an empty state F_j (where $R_j = L_j + 1$). Note interpolation search is identical to binary search except that it uses Eq. (1.1) rather than the formula $C_i = \lceil (L_i + R_i)/2 \rceil$ to define the address probed by the ith iteration.

The desirable aspect of interpolation search is that it runs in expected time $\log \log N$ when the set of keys are generated by the uniform distribution. Perl *et al.* (Perl, Itai, and Avni, 1978; Perl and Reingold, 1977) offer short proofs that the interpolation search has complexity $O(\log \log N)$, and (Gonnet, 1977; Yao and Yao, 1976) provide a slightly more elaborate treatment that establishes the tighter upper bound of $\log \log N + O(1)$. Yao and Yao (1976) also prove the interpolation search is optimal up to an additive constant by showing that no procedure can search unindexed ordered files in time better than $\log \log N - O(1)$. Willard (1981, 1983a) illustrates modifications of interpolation search that are useful for probability distributions that are **both nonuniform** and **unknown** to the search algorithm, and (Mehlhorn and Tsakalidis, 1985) presents several useful generalizations of the latter result. See also (Karlson, Munro, and Robertson, 1985; Van Emde Boas, Kass, and Zijlstra, 1977; Willard, 1983b, 1984) for some alternatives to interpolation search for indexed files.

Throughout this paper, N_i will denote the number of unprobed keys in the subfile $F_i = \{Y_{L_i} < Y_{L_i+1} < \cdots Y_{R_i}\}$; in other words,

$$N_i = R_i - L_i - 1. \tag{1.2}$$

Let P denote a fixed integer constant $\geqslant 2$, and define VARIATION(P) to be an algorithm that is identical to interpolation search except that if the iteration i has $N_i \leqslant P$ then VARIATION(P) will turn on P parallel processors and reach a termination during this iteration. One of our theorems generalizes methods from Yao and Yao (1976) and shows VARIATION(P) searches files generated by the uniform probability distribution in expected time $\log\log N - \log\log P + O(1)$. Intuition would suggest that even faster times would be possible if P parallel processors were used during each iteration of a search. However, the lower bound in Section 5 shows, to the contrary, that such further parallel processing improves performance by no more than an additive constant. (By an additive constant, this paper means a quantity whose value is, of course, independent of both N and P.)

The results summarized in the preceding paragraph are surprising because they take place in a complexity model, henceforth called the **iterative time** model, that measures costs by just counting the number of iterations of a SIMD machine searching the file F. Since this complexity model does not take into account the further time cost for communication delays, it is somewhat surprising that parallel algorithms using P processors during every iteration have only $O(1)$ fewer iterations than the essentially sequential VARIATION(P) procedure. An added surprise is that *nontrivial* parallel algorithms are demonstrably slower than VARIATION(P) *when one takes into account communication delays.* In particular for any $C > 1$, consider a cost model where a SIMD machine's time to search P addresses in parallel exceeds by any factor $C > 1$ the comparable search time of a nonparallel machine (because of communication delays). Certainly, one would expect the parallel SIMD machine to be slower if C was a large number, as say $C = P$. However, the surprising fact is that our theorem implies that the SIMD machine is also slower for any $C > 1$ because a multiplicative constant loss clearly outweighs an additive constant gain. That is, a fully parallel interpolation algorithm between P *communicating* processors *is inherently slower* because its number of iterations decreases by an additive constant whereas the communication cross-talk increases the cost of each iteration by a multiplicative constant.

Until Theorem 5.6 at the end of this paper, we will not again consider the cost of communication delays and our discussion will instead focus on the "iterative time" model, where communication is free. Our main goal will be to establish that VARIATION(P) falls within an additive constant of optimality under such tight models that ignore communication costs. This result will imply that VARIATION(P) does even better under the final models of Theorem 5.6, that take into account communication delays.

2. Intuition Behind the Proof

The results announced in the last section may sound counterintuitive, but the intuition behind their proof is actually quite simple. Let D_i be defined

$$\begin{aligned} D_i &= \text{MIN}(R_i - C_i^*,\ C_i^* - L_i) \\ &= \lceil \text{MIN}(Y_{R_i} - y,\ y - Y_{L_i}) \cdot N_i / (Y_{R_i} - Y_{L_i}) \rceil. \end{aligned} \tag{2.1}$$

A theme in the prior literature about sequential interpolation is that D_{i+1} satisfies the following equality with high probability

$$D_{i+1} \cong \sqrt{D_i}. \tag{2.2}$$

It is easy to verify that if Eq. (2.2) held with 100% certainty then $\log\log D_1$ iterations would be sufficient to reduce an initial file F_1 to a state j where $D_j \leqslant O(1)$. Gonnet (1977) and Yao and Yao (1976) establish that interpolation search runs in time $\log\log D_1 + O(1)$ by showing Eq. (2.2) holds with extremely high probability as D_i approaches infinity. Further, Yao and Yao (1976) show that no algorithm can run in better time than $\log\log D_1 - O(1)$ by establishing essentially that no algorithms can cause the expected value of D_{i+1} to shrink faster than Eq. (2.2).

Our analogous analysis will indicate that any algorithm using P parallel processors satisfies Eq. (2.3) with high probability:

$$D_{i+1} \geqslant \Omega(\sqrt{D_i}/P). \tag{2.3}$$

A reader can appreciate part of the intuition behind our lower bound $\log\log N - \log\log P - O(1)$ by noting that if Eq. (2.3) held with 100% probability and if $D_1 \cong N$ then clearly $\log\log N - \log\log P - O(1)$ iterations would be necessary to reach a state where $D_j \leqslant O(1)$.

On the other hand, Section 4 of our paper shows that a consequence of Yao and Yao (1976) and Gonnet, Rogers, and George (1980) is that $\log\log N - \log\log P + O(1)$ iterations are on the average sufficient for interpolation search to produce a state where $N_j \leqslant P$ when the file is generated by the uniform probability. A consequence of this fact is that the expected number of iterations for VARIATION(P) to reach a termination is $\log\log N - \log\log P + O(1)$. Hence, the combination of our upper and lower bounds shows that parallel processing helps by no more than an additive constant when applied outside the very last iteration of interpolation search!

Once again, it should be repeated that the results in the preceding paragraph hold *even when* one assumes a zero cost for communications between parallel processors, and VARIATION(P) will be proven by

Theorem 5.6 to *actually be faster* than any parallel algorithm *when any type of non-zero* communication delay appears.

The next three sections prove these facts formally by examing the exact probabilities associated with our search process. As with prior literature about interpolation search, the formal analysis is more complicated than an intuitive overview, where the probability qualifications are absent.

One further point. The theorems in the next two sections technically discuss probabilistic bounds for the case of unsuccessful searches (that is, the case where the target key y is different from the N stored keys $Y_1 < Y_2 < \cdots < Y_N$). The same time bounds would hold up to an additive constant for successful searches, but it is preferable to discuss only unsuccessful searches for the sake of simplicity. Many of the earlier papers about sequential interpolation search make a similar simplification (Gonnet, 1977; Yao and Yao, 1976), where they discuss either successful searches or unsuccessful searches, but not both.

3. Background Information From Probability Theory

This section provides the machinery from probability theory for analyzing parallel interpolation search. Let $B(j, N, p)$ denote the probability that N Bernoulli trials (Feller, 1968) produce j successes when each trial has a probability p of success. Then it is well known that

$$B(j, N, p) = \binom{N}{j} p^j (1-p)^{N-j}. \tag{3.1}$$

We now list three fairly easy consequences of classic probability that our paper will employ. The proofs of these propositions are kept brief because they are straightforward consequences of classic probability theory.

Proposition 3.1. $B(j, N, j/N) \leqslant O[\text{MIN}(1; \sqrt{N}/\sqrt{j(N-j)})]$.

Proof. Follows by substituting Stirling's formula (Eisen, 1969) into Eq. (3.1). Q.E.D.

Proposition 3.2. $B(j, N, p) \leqslant O[\text{MIN}(1; 1/\sqrt{p(1-p)N})]$.

Proof. It is easily verified that

$$B(j, N, p) \leqslant \text{MAX}\{B(\lceil pN \rceil, N, p); B(\lfloor pN \rfloor, N, p)\}.$$

Since $\lceil pN \rceil \neq pN \neq \lfloor pN \rfloor$ for most values of p and N, Proposition 3.1 does not technically determine the value for the two Bernoulli probabilities on

the right side of the equation above. However, it is clear that a minor discrepancy in rounding errors cannot change asymptotic magnitudes very much and that the preceding equation therefore implies $B(j, N, p) \leqslant O(B(\lceil pN \rceil, N, \lceil pN \rceil/N)) + O(B(\lceil pN \rceil, N, \lfloor pN \rfloor/N))$. The latter asymptote lies in $O\{\text{MIN}(1; \sqrt{1/p(1-p)N})\}$ by Proposition 3.1. Q.E.D.

PROPOSITION 3.3. *$\exists k > 0$, $\forall p$, $\forall N$ there is a probability $> k$ that at least $\lfloor pN \rfloor$ successes occur in a sequence of N Bernoulli trials where each individual trial produces a success with a probability $p > 0$.*

Proof Outline. Since pN is the expected number of successes during N Bernoulli trials, the inequality $pN \geqslant \lfloor pN \rfloor$ certainly suggests, at least intuitively, that a result similar to Proposition 3.3 should hold. We will not give a detailed proof of Proposition 3.3 here because similar results are known in the probability literature, and the next paragraph will instead outline how to infer Proposition 3.3 from Samuels (1965).

Equation (23) of Samuels (1965) offers a lower bound on the probability of at most $\lceil pN \rceil$ successes occurring in N Bernoulli trials. The mirror image of this formula, of course, provides a lower bound for the probability of at least $\lfloor pN \rfloor$ successes occurring in N trials, and it indicates that this probability is bounded below by $[1 - 1/(1 + Np)]^{Np}$. Now we observe the the preceding probability equals

$$\left(1 + \frac{1}{Np}\right)^{-Np} = e^{-Np \cdot \ln(1 + 1/Np)} \geqslant e^{-Np/Np} = e^{-1}.$$

Hence, one constant meeting the claim of Proposition 3.3 is $k = 1/e$. Q.E.D.

Let $F_i = \{Y_{L_i} < Y_{L_i+1} < \cdots < Y_{R_i}\}$ once again denote the subfile that is examined in the iteration of a search for the key y. Define the 5-tuple,

$$S_i = (L_i, R_i, Y_{L_i}, Y_{R_i}, y), \tag{3.2}$$

as the **state** of the ith iteration of this search. Intuitively, this tuple represents all the information known about the file $F_i = \{Y_{L_i} < Y_{L_i+1} < \cdots < Y_{R_i}\}$ at the beginning of the ith iteration of our search algorithms. Say X is a **state-dependent variable** iff it is a function of the 5-tuple S_i. Three examples of state-dependent variables are C_i^*, N_i, and D_i (from Eq. (1.1), (1.2), and (2.1)). The variable D_i will play an especially important role in the discussion which follows.

Let Q_i denote the number of **interior** keys in the file $F_i = \{Y_{L_i} < \cdots < Y_{R_i}\}$ that satisfy $Y_j < y$, and $\tilde{D}_i$ represent the quantity:

$$\tilde{D}_i = \text{MIN}(Q_i, N_i - Q_i). \tag{3.3}$$

Say a variable is **file-dependent** iff its value is a function of the tuple $\{y, L_i, R_i, Y_{L_i}, Y_{L_i+1} \cdots Y_{R_i}\}$. Q_i and $\tilde{D}_i$ are examples of variables that are file-dependent but **not** state-dependent.

Assume X is a file-dependent variable. Let $\text{Prob}(X \leqslant j \mid S_i)$ denote the probability that $X \leqslant j$ assuming the interior keys in the file $F_i = \{Y_{L_i} < \cdots < Y_{R_i}\}$ are generated by the uniform distribution. Then this quantity is a state-dependent variable! Also X's expected value, denoted $E(X \mid S_i)$, is state-dependent.

The distinction between state- and file-dependent variables will be greatly helpful in providing a simple but rigorous analysis for parallel interpolation search. A variable can be curiously state-dependent relative to one iteration but file-dependent relative to another. For example, D_{i+1} is a state-dependent relative to S_{i+1}, but it is a file-dependent variable relative to S_i under all the search algorithms considered in this paper.

We will now state five lemmas about file-dependent variables. Some readers may wish to skim the proofs of these lemmas, since each is fairly straightforward consequence of Propositions 3.1 thru 3.3. The lemmas below are important because they provide the machinery for calculating the upper and lower bounds in Sections 4 and 5. In each of the next lemmas and throughout the paper, we assume the file F and therefore all the subfiles F_i are generated by the uniform probability distribution. For the sake of keeping their proposition statements short, *this assumption does not appear explicitly in the lemmas that follow.*

LEMMA 3.4. *The file-dependent variable Q_i satisfies the following three conditions:*

(A) *For each* $j \geqslant 0$ $\text{Prob}(Q_i = j \mid S_i) = B(j, N_i, (y - Y_{L_i})/Y_{R_i} - Y_{L_i}))$

(B) *If* $y \leqslant (Y_{L_i} + Y_{R_i})/2$ *then* $E(Q_i \mid S_i) \leqslant D_i$

(C) *For each* $j \geqslant 0$, $\text{Prob}(Q_i = j \mid S_i) \leqslant O(1/\sqrt{D_i})$.

Proof of Assertion A. Obvious; since the interior keys in F_i are selected by generating N_i records (initially in unsorted order), where each record's probability of being less than y is independent of the next record's likelihood of satisfying this condition, and this probability equals the quantity p defined

$$p = (y - Y_{L_i})/(Y_{R_i} - Y_{L_i}) \tag{3.4}$$

Proof of Assertion B. It is well known that $N_i \cdot p$ is the expected number of successes when p is the probability of success during a sequence of N_i Bernoulli trials. This quantity is $\leqslant D_i$ when $y \leqslant (Y_{L_i} + Y_{R_i})/2$, by Eqs. (2.1) and (3.4).

Proof of Assertion C. Equations (2.1) and (3.4) imply $\text{MAX}(1; p(1-p)N_i) \cong D_i$. Assertion (C) follows by substituting this inequality into Propositions 3.2 and 3.4A. Q.E.D.

LEMMA 3.5. *The file-dependent variable* $\tilde{D}_i$ *satisfies two conditions*:

(A) $E(\tilde{D}_i | S_i) \leqslant D_i$.

(B) $\text{Prob}(\tilde{D}_i \leqslant 2D_i | S_i) \geqslant 1/2$.

Proof of Assertion A. Since $\tilde{D}_i \leqslant Q_i$, it follows that

$$E(\tilde{D}_i | S_i) \leqslant E(Q_i | S_i). \tag{3.5}$$

The combination of Lemma 3.4B and Eq. (3.5) imply $E(\tilde{D}_i | S_i) \leqslant D_i$ whenever $y \leqslant (Y_{L_i} + Y_{R_i})/2$. By a symmetry argument, the same result must be true when $y \geqslant (Y_{L_i} + Y_{R_i})/2$.

Proof of Assertion B. Let us assume for the sake of contradiction that Assertion B is false, and therefore that $\tilde{D}_i$ had a probability $> \frac{1}{2}$ of exceeding $2D_i$. Since $\tilde{D}_i$ is always $\geqslant 0$, this would imply $E(\tilde{D}_i | S_i) > D_i$. But the latter inequality contradicts Assertion A, whose proof was given in the previous paragraph. Hence, we must conclude that Assertion B is valid because Assertion A was.

LEMMA 3.6. *There exists* $k > 0$ *such that every iteration* i *of interpolation search satisfies* $\text{Prob}(N_{i+1} \leqslant D_i | S_i) > k$.

Proof. With no loss of generality, we may assume $y \geqslant (Y_{L_i} + Y_{R_i})/2$. Then since $Q_i \geqslant N_i - D_i$ holds exactly when $N_{i+1} \leqslant D_i$, our formalism implies

$$\begin{aligned} \text{Prob}(N_{i+1} \leqslant D_i | S_i) &= \text{Prob}(Q_i \geqslant N_i - D_i | S_i) \\ &= \sum_{j=N_i-D_i}^{N_i} B(j, N_i, (y - Y_{L_i})/(Y_{R_i} - Y_{L_i})) \\ &\qquad \text{by Lemma 3.4A.} \end{aligned} \tag{3.6}$$

Proposition 3.3 then implies there exists some positive constant k, independent of N_i and D_i which bounds from below the latter term. (In particular, you can confirm that proposition 3.3 implies the last term in Eq. (3.6) is bounded below by k by setting $N = N_i$ and $p = (y - Y_{L_i})/(Y_{R_i} - Y_{L_i})$ and by noticing that Eq. (1.1), (1.2), and (2.1) imply that j in Eq. (3.6) is no greater than $\lfloor pN \rfloor$.) Q.E.D.

Define an **iterative search procedure with P parallel processes** as a algorithm which probes the subfile $F_i = \{Y_{L_i} < \cdots < Y_{R_i}\}$ by first

generating P indices $C_i^1 \leqslant C_i^2 \leqslant \cdots \leqslant C_i^p$ and then checking for whether any of these indices C_i^a has $Y_{C_i^a} = y$. If the answer is yes then the parallel algorithm terminates; otherwise using a notation convention where $C_i^0 = L_i$ and $C_i^{p+1} = R_i$, the next iteration will search the unique subfile of the form $\{Y_{C_i^a} < \cdots < Y_{C_i^{a+1}}\}$ whose range contains y. As with an interpolation search, this procedure continues until either the key y is found or the search space becomes empty, i.e., $N_i = 0$. (Note by Eq. (2.1), this condition also implies $D_i = 0$.)

LEMMA 3.7. *For each integer $m \geqslant 0$, every iterative search algorithm using P parallel processors satisfies* $\text{Prob}(D_{i+1} < m \mid S_i) \leqslant O(mP/\sqrt{D_i})$.

Proof. Lemma 3.4C implies each cut index C_i^a satisfies

$$\text{Prob}(|C_i^a - Q_i - L_i| < m \mid S_i) \leqslant O(m/\sqrt{D_i}) \tag{3.7}$$

The definition of $\tilde{D}$ (i.e., Eq. (3.3)) implies

$$\tilde{D}_{i+1} = \text{MIN}[|C_i^0 - Q_i - L_i|, |C_i^1 - Q_i - L_i|, \ldots, |C_i^{P+1} - Q_i - L_i|]. \tag{3.8}$$

These two equations imply

$$\text{Prob}(\tilde{D}_{i+1} < m \mid S_i) \leqslant O(mP/\sqrt{D_i}). \tag{3.9}$$

Equation (3.9) implies there exists some constant $k > 0$ such that

$$\text{Prob}(\tilde{D}_{i+1} < m \mid S_i) \leqslant kmP/\sqrt{D_i}. \tag{3.10}$$

Applying Lemma 3.5B to the latter equation we conclude

$$\text{Prob}(D_{i+1} < m/2 \mid S_i) \leqslant 2kmP/\sqrt{D_i}. \tag{3.11}$$

Since **all** numbers $m > 0$ satisfy Eq. (3.11), so must **all** numbers $m > 0$ satisfy

$$\text{Prob}(D_{i+1} < m \mid S_i) \leqslant 4kmP/\sqrt{D_i} \leqslant O(mP/\sqrt{D_i}). \qquad \text{Q.E.D.}$$

In the remainder of this article, $\log\log^+ j$ denotes $\max(1, \log\log j)$, and $\text{LSUM}(j)$ denotes the quantity

$$\text{LSUM}(j) = (1/j) \sum_{i=0}^{j-1} \log\log^+ i. \tag{3.12}$$

This definition implies:

LEMMA 3.8. $\text{LSUM}(j) \geqslant \log\log^+ j - O(1/\log j)$.

Proof. Let $\text{COUNT}(k, j)$ denote the number of distinct non-negative integers i, where $(\log\log^+ j - \log\log^+ i)$ has a value lying between $(k-1)/\log j$ and $k/\log j$. It is easily seen that $\text{COUNT}(k, j) \leqslant O(j2^{-k})$. The proof of Lemma 3.8 is then completed by the observation

$$\log\log^+ j - \text{LSUM}(j) \leqslant \frac{1}{j\log j}\sum_{k=1}^{\infty} k \cdot \text{COUNT}(k, j) \leqslant O\left(\frac{1}{\log j}\right). \quad \text{Q.E.D.}$$

The next section shows how Lemmas 3.6 and Gonnet *et al.* (1980) and Yao and Yao (1976) imply the $\log\log N - \log\log P + O(1)$ iterative complexity of VARIATION(P), and Section 5 shows how to establish a complementary lower bound by using Lemmas 3.7 and 3.8.

4. ANALYSIS OF VARIATION(P)

This section proves VARIATION(P) runs in expected iterative time $\log\log N - \log\log P + O(1)$. Our discussion begins with one preliminary lemma whose proof explains how our new result is related to the prior literature.

LEMMA 4.1. *Suppose interpolation search is probing a file whose initial state satisfies $D_1 \geqslant d$. Then the expected number of iterations to reach a state j satisfying $D_j < d$ is $\leqslant \log\log N - \log\log d + O(1)$ when the file is generated by the uniform probability distribution.*

Proof. Let L_d^+ denote the greatest lower bound on the expected number of iterations for interpolation search to reach a terminating $D = 0$ state when it starts in a state j satisfying $D_j > d$. Also, let L_d^- denote the greatest lowest bound on the expected number of additional iterations that interpolation search will need to finish the same search *just after* it has reached a state satisfying $D_j \leqslant d$. The phrase "just after" in the definition of L_d^- implies that this lower bound has the relationship to L_d^+,

$$L_d^- \geqslant L_d^+ - 1. \tag{4.1}$$

Also, let $U(N)$ denote a least upper bound on the expected number of iterations for interpolation search to reach the terminating $D = 0$ state during a search on a file of size N. This formulation is useful because it enables us to apply Gonnet *et al.* (1980) and Yao and Yao (1976) to prove Lemma 4.1. In particular, Eq. (4.2) is the main upper bound theorem of

their works and Eq. (4.3) is the main lower bound theorem of Yao and Yao (1976)[1]:

$$U(N) \leqslant \log\log N + O(1), \tag{4.2}$$

$$L_d^+ \geqslant \log\log d - O(1). \tag{4.3}$$

These results imply the validity of Lemma 4.1, since the expected time to reach a state S_j, where $D_j \leqslant d$ is then certainly no more than $U(N) - L_d \leqslant U - L_d^+ + 1 \leqslant \log\log N - \log\log d + O(1)$. Q.E.D.

THEOREM 4.2. *The expected number of iterations used by the algorithm* VARIATION(P) *is* $\log\log N - \log\log P + O(1)$ *when the file is generated by the uniform probability distribution.*

Proof. Yao and Yao (1976) show the D-vector forms a nonincreasing sequence over time; therefore if $D_j \leqslant P$ then all $i \geqslant j$ satisfy $D_i \leqslant P$. In this context, Lemma 3.6 implies that $O(1)$ is the expected number of iterations for VARIATION(P) to reach a state satisfying $N_i \leqslant P$ **after** it has reached a state j satisfying $D_j \leqslant P$.

But Lemma 4.1 indicates that $\log\log N - \log\log P + O(1)$ is the expected number of steps needed to satisfy $D_j \leqslant P$. Taking the sum of these two quantities, we see that $\log\log N - \log\log P + O(1)$ bounds the expected number of steps for VARIATION(P) to reach a termination. Q.E.D.

5. THE MATCHING LOWER BOUND

Yao and Yao give a lower bound for the expected time of a sequential search into a random ordered table, which we will now generalize to any algorithm using P parallel processors.

Let SET(d) denote the set of all file states S_i satisfying $D_i \geqslant d$ and ALG(P) the set of all iterative search algorithms that use no more than P processors during each iteration. If $S \in \text{SET}(d)$ and $A \in \text{ALG}(P)$ then TIME(A, S) shall denote the expected time that the algorithm A needs to process a state S when it is searching a uniformly generated file. Define $T_P(d)$ as the greatest lower bound on TIME(A, S) for the set of all ordered pairs (A, S) where $A \in \text{ALG}(P)$ and $S \in \text{SET}(d)$. Two immediate consequences of this definition are

$$T_P(0) = 0 \tag{5.1}$$

$$\text{if} \quad d_1 \leqslant d_2 \qquad \text{then} \quad T_P(d_1) \leqslant T_P(d_2). \tag{5.2}$$

We will now use these recurrence equations to prove the following lemma.

[1] Readers who wish to examine (Yao and Yao, 1976) should note that our quantity D corresponds to their $\lceil N \cdot \text{MIN}(\alpha, 1-\alpha) \rceil$, and that their term α corresponds to our ratio $(y - Y_L)/(Y_R - Y_L)$.

LEMMA 5.1. *There exists a constant $k > 1$ whose value is independent of both P and d that assures*

$$T_P(d) \geq 1 + \frac{1}{\lceil \sqrt{d}/(kP) \rceil} \sum_{j=0}^{\lceil \sqrt{d}/(kP) \rceil - 1} T_P(j). \tag{5.3}$$

Proof. Let $\text{Prob}(A, F, j)$ denote the probability that the $(i+1)$th iteration of the algorithm A will have $D_{i+1} = j$ when the ith iteration of this algorithm is searching a subfile F. Also, let $T_{A,P}(d)$ denote a lower bound on A's remaining expected number of iterations when the ith iteration has $D_i = d$ and A is using P parallel processors. Then our notation suggests

$$T_{A,P}(d) \geq 1 + \sum_{j=0}^{\infty} \text{Prob}(A, F, j) \cdot T_{A,P}(j) \geq 1 + \sum_{j=0}^{\infty} \text{Prob}(A, F, j) \cdot T_P(j). \tag{5.4}$$

Theorem 3.7 implies every algorithm with P parallel processors must satisfy $\text{Prob}(A, F, j) \leq O(1/P\sqrt{d})$, and this fact indicates that there must exist a constant k, whose value is independent of both P and d such that

$$\text{Prob}(A, F, j) \leq 1/\sqrt{d}(kP). \tag{5.5}$$

Since $T_P(1) \leq T_P(2) \leq T_P(3)$..., Eqs. (5.4) and (5.5) imply

$$T_{A,P}(d) \geq 1 + \frac{1}{\lceil \sqrt{d}\, kP \rceil} \sum_{j=0}^{\lceil \sqrt{d}\, kP \rceil - 1} T_P(j). \tag{5.6}$$

Since $T_{A,P}(d) = T_P(d)$ when A is chosen to be the fastest possible search algorithm in the set $\text{ALG}(P)$, Eq. (5.6) implies (5.3). Q.E.D.

The chief goal of this section is to show recurrence relations (5.1) thru (5.3) imply $T_P(d) \geq \log\log d - \log\log P - O(1)$. Recall that LSUM was defined in Eq. (3.12). We state one preliminary lemma about this quantity before proving our main result.

LEMMA 5.2. *There exist constants $C > 0$ and $C^* > 0$ independent of both i and j, such that if $j \geq i^2 > 1$ then the following two equations hold:*

$$\log\log(j/i) \geq \log\log(j) - C^* \log(i)/\log(j/i) \tag{5.7}$$

$$\text{LSUM}(\lceil j/i \rceil) \geq \log\log(j) - C \cdot \log(i)/\log(j/i). \tag{5.8}$$

Proof. Let a and b be a pair of positive numbers satisfying $a < b$. Since

the natural log function has a derivative $\leqslant 1/a$ over the interval $[a, b]$, it must follow that

$$\ln(b) \leqslant \ln(a) + \frac{b-a}{a}. \tag{5.9}$$

Throughout this paper, we take logarithms in base 2 rather than in base e. If we let C^* denote the constant $1/\ln 2$, then the immediate translation of Eq. (5.9) into base 2 logarithms is

$$\log(b) \leqslant \log(a) + C^* \frac{b-a}{a}. \tag{5.10}$$

If we set $b = \log j$ and $a = \log(j/i)$, Eq. (5.10) implies

$$\log\log j \leqslant \log\log(j/i) + C^* \frac{\log j - \log(j/i)}{\log(j/i)}. \tag{5.11}$$

Equation (5.11) clearly implies Eq. (5.7) by the fact that $\log i = \log j - \log(j/i)$. Equation (5.8) is then an immediate consequence of Eq. (5.7) and Lemma 3.8. Q.E.D.

Now we prove a proposition whose corollary is that $T_P(d) \geqslant \log\log d - \log\log P - O(1)$.

THEOREM 5.3. *Let $C > 0$ and $k > 1$ denote the constants defined in Lemmas* 5.1 *and* 5.2. *Then*

$$T_P(d) \geqslant \log\log d - \log\log(kP) - 2 - C \cdot \mathrm{MAX}\{0; 2 - 2\log(kP)/\log d\}. \tag{5.12}$$

Proof by Induction on d. It is obvious Eq. (5.12) is valid when $d \leqslant (kP)^4$, since the right side of (5.12) is then $\leqslant 0$. For the case of $d > (kP)^4$, we may inductively assume Eq. (5.12) holds for all $d^* < d$. Then the combination of this assumption and Eq. (5.3) implies

$$\begin{aligned} T_P(d) \geqslant 1 + \mathrm{LSUM}(\lceil \sqrt{d}/(kP) \rceil) - \log\log(kP) - 2 \\ - C \cdot \mathrm{MAX}\{0; 2 - 2\log(kP)/\log(\sqrt{d}/kP)\}. \end{aligned} \tag{5.13}$$

By substituting Eq. (5.8) and the facts $kP > 1$ and $d > (kP)^4$ into the inequality above, we get the desired conclusion that d also satisfies Eq. (5.12). Q.E.D.

COROLLARY 5.4. $T_P(d) \geqslant \log\log d - \log\log P - O(1)$.

Proof. Since C and k were constants in Theorem 5.3 and since we always assume $P \geqslant 2$ when discussing parallel processing, it is evident that both $|\log\log(kP) - \log\log P|$ and $C \cdot \text{MAX}[0; 2 - 2\log(kP)/\log d]$ are also bounded by constants independent of both P and d, i.e., they are both $O(1)$. From those facts, it is evident that Eq. (5.12) implies Corollary 5.4. Q.E.D.

We now state a second corollary which indicates that VARIATION(P) is optimal up to an additive constant.

COROLLARY 5.5. *Suppose the records in the ordered file* $F_1 = \{Y_1 < Y_2 \cdots Y_N\}$ *are generated by the uniform distribution over* (0, 1) *and this distribution also generates the search key* y. *Then* $\log\log N - \log\log P - O(1)$ *lower bounds the expected number of iterations to search the file* F_1 *with any algorithm using* P *or fewer parallel processors.*

Proof. For simplicity, our proof shall assume N is an even number. Then since y is generated by the uniform distribution, it follows that $\text{Prob}(D_1 = i) = 2/N$, for each $i \leqslant N/2$. Let $\text{LOW}_P(N)$ denote a lower bound on the iterative time for an algorithm to search a file of size N. Then the last two sentences combined with Corollary 5.4 imply

$$\begin{aligned}\text{LOW}_P(N) &\geqslant \frac{2}{N}\sum_{d=1}^{N/2} T_P(d) \\ &\geqslant \text{LSUM}(N/2) - \log\log P - O(1) \\ &\geqslant \log\log N - \log\log P - O(1) \qquad \text{by Lemma 3.8.}\end{aligned}$$

Q.E.D.

The key phrase in Corollary 5.5 was that $\log\log N - \log\log P - O(1)$ lower bounded "the expected number of iterations" of the file search. In Corollary 5.5, as well as in all our other propositions, we ignored the cost of communication delays. Even in this tight model, we have seen the combination of Theorem 4.2's upper bound and Corollary 5.5's lower bound imply that VARIATION(P) falls within an additive constant of optimality. Our next theorem shows VARIATION(P) compares even better to its alternatives in models that take into consideration communication delays.

THEOREM 5.6. *Assume that each iteration of* VARIATION(P) *takes one unit of time, and let* A *denote an algorithm using* P *parallel processors where each iteration consumes* c *units of times (when one takes into account communication delays). Then if* $c > 1$, *the parallel algorithm* A *will be slower than* VARIATION(P), *as the problem size* N *attains a limiting value approaching infinity.*

Proof. Proposition 4.2 and Corollary 5.5 imply respectively that there exist two constants $K_1 > 0$ and $K_2 > 0$ such that Eq. (5.14) and (5.15) characterize the cost of these two algorithms *when* communications delays are considered:

$$\text{VARIATION}(P)\text{'s TIME} \leqslant \log\log N - \log\log P + K_1 \qquad (5.14)$$

$$\text{LOWER BOUND ON } A\text{'s TIME} \geqslant c \cdot (\log\log N - \log\log P - K_2). \qquad (5.15)$$

Since K_1 and K_2 are constants and since the hypothesis of Theorem 5.6 indicated $c > 1$, these equations imply that for fixed P, VARIATION(P)'s real time will be better than A's actual time as N approaches infinity.

Q.E.D.

6. Conclusion

We have determined, within an additive constant, the expected complexity of parallel searching an unindexed ordered random table with keys independently chosen from a uniform distribution, and we have demonstrated that parallel processing does not significantly improve the complexity of this search problem except when used in the very last iteration. Our theorems hold in even the most conservative case where the communication delay is zero. Theorem 5.6 shows that our proposed VARIATION(P) procedure becomes even more cost-effective in models where the communication delay is assumed to have a nonzero cost.

Acknowledgment

We thank the referee for his useful suggestions for improving the presentation.

Received May 5, 1988; accepted July 25, 1988

References

Atjai, M., Fredman, M. L., and Komlos, J. (1983), Hash functions for priority queues, *in* "24th IEEE Symp. on Found. of Comput. Sci." pp. 299–303.

Eisen, M. (1969), "Introduction to Mathematical Probability Theory," Prentice–Hall, Englewood Cliffs, NJ.

Feller, W. (1968), "An Introduction to Probability Theory and Its Applications," Vol. I, 3rd ed., New York.

Gonnet, G. H. (1977), "Interpolation and Interpolation-Hash Searching," Ph.D. thesis, University of Waterloo.

GONNET, G. H., ROGERS, L. D., AND GEORGE, J. A. (1980), An algorithmic and complexity analysis of interpolation search, *Acta Inform.* **13**, 39–52.

KARLSON, R. G., MUNRO, J. I., AND ROBERTSON, E. L. (1985), The nearest neighbor problem on bounded domains, *in* "12th ICALP," Lect. Notes in Comput. Sci. Vol.. 196, pp. 318–327, Springer-Verlag, New York/Berlin.

KARLIN, S., AND TAYLOR, H. M. (1975), "A First Course in Stochastic Processes," 2nd ed., Academic Press, New York.

KNUTH, D. E. (1973), "The Art of Computing Programming, Vol. 3: Sorting and Searching," Addison-Wesley, Reading, MA.

KRUIJER, H. S. M. (1974), The interpolated file search method, *Informatie* **16**, 612–615.

MEHLHORN, K., AND TSAKALIDIS, A. (1985), Dynamic interpolation search, *in* "Proceedings, 12th ICALP, Lect. Notes in Comput. Sci. Vol. 196, pp. 424–434, Springer-Verlag, New York/Berlin.

PERL, Y., AND REINGOLD, E. M. (1977), Understanding and complexity of interpolation search, *Inform. Process Lett.* **6**, 219–222.

PERL, Y., ITAI, A., AND AVNI, H. (1978), Interpolation search-A $\log \log N$ search, *Comm. ACM* **21**, 550–557.

PETERSON, W. W. (1957), Addressing for random-access storage, *IBM J. Res. Develop.* **1**, 130–146.

SAMUELS, S. M. (1965), On the number of successes in independent trials, *Ann. of Math. Statist.* **36**, 1272–1278.

VAN EMDE BOAS, P., KASS, R., AND ZIJLSTRA, E. (1977), Design and implementation of an efficient priority queue, *Math. Systems Theory* **10**, 99–127.

WILLARD, D. E. (1981/85), A $\log \log N$ search algorithm for nonuniform distributions, Extended abstract in "Proceedings, ORSA-TIMS Conference on the Applied Probability-Computer Science Interface," Vol. II, pp. 3–14, 1981; full length, *SIAM J. Comput.* 1013–1029.

WILLARD, D. E. (1983a), Surprisingly efficient search algorithms for nonuniformly generated files, *in* "21st Allerton Conference on Communications, Control and Computing," pp. 656–662.

WILLARD, D. E. (1983b), Log-logarithmic worst-case range queries are possible in space $O(N)$, *Inform. Process. Lett.* **17**, 81–84.

WILLARD, D. E. (1984), New trie data structures which support very fast search operations, *J. Comput. System Sci.* **28**, 379–394.

YAO, A. C., AND YAO, F. F. (1976), The complexity of searching an ordered random table, *in* "Proceedings. 17th Annual Symposium on Foundations of Computer Science," pp. 173–177.