Int. J. Communications, Network and System Sciences, **2011, 4, 549-561**
doi:10.4236/ijcns.2011.49066 Published Online September 2011 (http://www.SciRP.org/journal/ijcns)

Parallel Minimax Searching Algorithm for Extremum of Unimodal Unbounded Function

Boris S. Verkhovsky
Department of Computer Science, New Jersey Institute of Technology, Newark, USA
E-mail: *verb73@gmail.com*
Received June 27, 2011; *revised August* 1, 2011; *accepted August* 25, 2011

Abstract

In this paper we consider a parallel algorithm that detects the maximizer of unimodal function $f(x)$ computable at every point on unbounded interval $(0,\infty)$. The algorithm consists of two modes: scanning and detecting. Search diagrams are introduced as a way to describe parallel searching algorithms on unbounded intervals. Dynamic programming equations, combined with a series of liner programming problems, describe relations between results for every pair of successive evaluations of function *f* in parallel. Properties of optimal search strategies are derived from these equations. The worst-case complexity analysis shows that, if the maximizer is located on a priori unknown interval $(n-1,n]$, then it can be detected after $c_p(n)=\left\lceil 2\log_{\lceil p/2\rceil+1}(n+1)\right\rceil-1$ parallel evaluations of $f(x)$, where *p* is the number of processors.

Keywords: Adversarial Minimax Analysis, Design Parameters, Dynamic Programming, Function Evaluation, Optimal Algorithm, Parallel Algorithm, System Design, Statistical Experiments, Time Complexity, Unbounded Search, Unimodal Function

1. Introduction and Problem Statement

Design of modern systems like planes, submarines, cars, aircraft carriers, drugs, space crafts, communication networks etc. is an expensive and time consuming process. In many cases the designers must run and repeat expensive experiments, while each run is a lasting process. The goals of these experiments can be either to maximize performance parameters (speed, carried load, degree of survivability, healing effect, reliability etc.) or to minimize fuel consumption, undesirable side effects in drugs, system's cost etc.

Performance parameters that are to be maximized (or minimized) are functions of other design parameters (wing span in aircrafts, aerodynamic characteristics of cars, planes, helicopters, raising cars, antennas or hydrodynamic profiles of submarines, ships etc.). At the best, these functions are computable if CAD is applicable. Otherwise, multitude of wind tunnel experiments is required for aero- or hydrodynamic evaluations. Analogously, numerous statistical experiments on animals and later on different groups of people are necessary if a new drug is a subject of design. Such experiments may last months or even years. For instance, in the USA development of a new drug takes in average ten-twelve years. In view of all factors listed above, it is natural to minimize the number of experiments in attempts to design a system with the best parameters. In most of the cases, we can estimate an upper bound on the design parameter under consideration; yet this is not always the case especially if the cost of experiments is increasing or even can be fatal in design of new drugs.

The unbounded search problem, as described in [1], is a search for a key in a sorted unbounded sequence. The goal of the optimal unbounded search is to find this key for a minimal number of comparisons in the worst case. The authors describe an *infinite* series of sequential algorithms (*i.e.*, using a single processor), where each algorithm is more accurate, than the previous algorithm. In [2] the unbounded search problem is interpreted as the following two-player game. Player A chooses an arbitrary positive integer n. Player B may ask whether the integer x is less than n. The "cost" of the searching algorithm is the number of guesses that B must use in order to determine n. The goal of the player B is to use a minimal number of these guesses in the worst case.

This number is a function $c(n)$. The author of that paper provides lower and upper bounds on the value of $c(n)$. More results on nearly-optimal algorithms are provided in [3], and then these results are generalized for a transfinite case in [4].

As pointed out in [5], the problem formulated in [2] is equivalent to the search for a maximizer of an unimodal function $f(x)$, where $x \in (0, \infty)$. The goal of the search is to minimize the number of required evaluations of a function $f(x)$ (*probes*, for short) in the worst case, if the maximizer is located on a priori unspecified interval $(n-1, n]$, and where n is a positive integer number. In [5], the authors consider the unbounded discrete unimodal *sequential* search for a maximizer. Employing an elaborate apparatus of Kraft's inequality, [5], inverse Fibonacci Ackermann's function and, finally, a repeated diagonalization, they construct a series of algorithms that eventually approach lower bounds on the function $c(n)$.

The general theory of optimal algorithms is provided in [6,7]. The problems where f is a unimodal function, defined on a *finite* interval, are analyzed by many authors, and the optimal algorithms are provided and analyzed in [8-12]. Optimal parallel algorithms, searching for a maximum of a unimodal function on a *finite* interval, are discussed in [13-15]. The case where f is a *bimodal* function is discussed and analyzed in [16-18]. The case where additional information is available is studied in [19,20]. The optimal search algorithm for the maximum of a unimodal function on a finite interval is generalized for a case of multi-extremal function in [21-23]. In all these papers, the optimal algorithms are based on the mere fact that a maximizer (minimizer) is located on *a priori known finite* interval K (called the interval of uncertainty). The algorithms employ a procedure that shortens the interval of uncertainty K after every probe. Complexities of related problems are functions of the size of interval K. Search algorithms for two-dimensional and multidimensional unimodal functions are considered respectively in [24,25].

In this paper we consider a parallel algorithm finding a maximizer (or a minimizer) of a function f defined on an *unbounded* interval I of $\boldsymbol{R}$ and computable at every point $x \in I$. Without loss of generality, we assume that $I = (0, \infty)$. It is easy to see that an algorithm, that detects a maximizer, cannot employ the same or analogous strategies as in the finite case, since the interval of uncertainty is infinite.

Definition 1.1. A unimodal function has the following properties: 1) there exists a positive number s, such that $f(x_1) < f(x_2) < f(s)$ for all $0 \le x_1 < x_2 < s$ and $f(s) > f(x_1) > f(x_2)$ for all $s < x_1 < x_2 < \infty$; 2) $f(x)$ is not constant on any subinterval of I. The point s is called a *maximizer* of function $f(x)$. It is not required that f be a smooth or even a continuous function.

The goal of this paper is to describe and analyze an algorithm that 1) detects an interval of length *t* (*t*-interval, for short) within which a maximizer of f is located; 2) uses a minimal number of parallel probes (*p-probes*, for short) in the worst case for the *t*-detection.

Definition 1.2. An algorithm is called *balanced* if it requires an equal number of probes for both stages (scanning and detection).

Definition 1.3. The algorithm that is described in this paper is minimax (optimal) in the following sense. Let *F* be a set of all unimodal functions $f \in F$ defined on *I*; S_t be a set of all possible strategies s_t detecting a *t*-interval that contains a maximizer *s* of function *f*; and let $N(f, s_t)$ be the number of *p*-probes that are required for detection of the maximizer on *t*-interval using strategy s_t. Then a minimax strategy s_t^o detects the maximizer for a minimal number of p-probes in the worst case of the unbounded function *f*, [17].

The Definition 1.3 implies that

$$N\left(s_t^o\right) = \min_{s_t \in S_t} \max_{f \in F} N\left(f, s_t\right) \tag{1.1}$$

Remark 1.1. Although *s* is a priori unknown to the algorithm designer, it is assumed in this paper that its value is *fixed*. Otherwise, the algorithm designer will not be able to provide any algorithm for *t*-detection of *s*. Indeed, the adversary can generate a function *f* that is increasing on any finite subinterval $(0, v) \subset I$.

2. Choice of Next Evaluation Point

2.1. Sequential Search: Single-Processor Case

Proposition 2.1. Let us consider two arbitrary points, *L* and *R*, that satisfy inequalities $0 < L < R < \infty$. If $f(L) < f(R)$, then maximizer *s* is greater than *L*; if $f(L) > f(R)$ then maximizer *s* is smaller than *R*; if $f(L) = f(R)$, then *s* is greater than *L* and smaller than *R*, *i.e.*, $s \in (L, R)$, [10,26].

Proof follows immediately from unimodality of the function *f*.

If $f(L) \ge f(R)$, then a maximizer *s* is detected on a finite interval, *i.e.*, $s \in (0, R)$. Therefore for *t*-detection of the maximizer *s* we can employ Kiefer's algorithm for sequential search [10,26] or the algorithm [25] for parallel search.

Suppose that *f* is evaluated at two points $q_i := L$ and $q_j := R$, where $0 < L < R < \infty$ and let $f(L) < f(R)$. Two strategies are possible in this case: to evaluate *f* either at a point $M \in (L, R)$ or at a point $M > R$; {there is no reason to evaluate *f* at $q < L$, since the Proposition 2.1 guarantees that if $f(L) < f(R)$, then maximizer *s* is greater than *L*}. Let assume that function *f* is increasing on interval $(L, R-\varepsilon)$.

Therefore, s is either on finite interval $(R-\varepsilon, R)$ or on infinite interval $R \le s < \infty$. Keeping in mind that we consider the worst case complexity, it is reasonable to evaluate f at a point $M > R$ that is on the infinite interval $[R, \infty)$.

2.2. Multiprocessor Case

Let us consider the same function $f(x)$ as in the single processor case. Let us simultaneously evaluate it at points $M_1, \cdots, M_p$, where p is the number of available processors.

Consider four scenarios:

A: All or a part of the points $M_1, \cdots, M_p$ are *inside* interval (L, R);

B: All points $M_1, \cdots, M_p$ are *outside* interval (L, R), *i.e.*, every point $M_1, \cdots, M_p$ is larger than R;

C: $s \in [R-\varepsilon, R]$;

D: $s \notin [R-\varepsilon, R]$.

2.3. Possible Outputs in the Worst Case

For both AC and AD scenarios $f(x)$ and ε can be selected in such a way that for all $1 \le i \le p$, for which $M_i \in (L, M)$, the inequality $f(M_i) < f(R)$ holds. Hence, taking into account that we are dealing with the worst case, all evaluations must be done outside interval (L, R) if $f(L) < f(R)$.

3. Optimal Unbounded Search Algorithm as a Two-Player Game with Referee

3.1. Sequential Search: (p = 1)

Let us consider two players A and B and a referee. Their game consists of two stages.

- At *the beginning*, player A selects a value $s > 0$ and informs the referee about it. Player B selects a value t > 0 and informs player A and the referee about his choice.
- At *the first stage*, B sequentially selects positive and distinct points $q_1, q_2, \cdots, q_i$. The referee terminates the first stage if there are points q_j and q_k such that $s < q_j$ and $s < q_k$.
- *The second stage* begins from state (u_1, v_1, w_1) where $u_1 := q_{j-1}$; $v_1 := q_j$; $w_1 := q_{j+1}$.

At the second stage, B selects points and A selects intervals. Let the game be in state (u_j, v_j, w_j) of the second stage. This stage terminates if $|w_j - u_j| < t$. Otherwise B selects a point $x_j \ne v_j$, such that $u_j < x_j < w_j$. Then A eliminates the leftmost or the rightmost subinterval, [16]. The goal of player B is to minimize the number of points required to terminate the game. The goal of player A is to maximize the number of these points. The adversarial approach for interpretation of the optimal search algorithms is also considered in [2,27].

Remark 3.1. It is easy to see that B is an algorithm designer and A is a user that selects function f and required accuracy t.

3.2. Multiple-Processor Search: ($p \ge 2$)

An optimal parallel search algorithm with p processors has an analogous interpretation. In this case, at the first phase of the game, player B on his move selects p distinct positive points. The referee *terminates the first phase* if there are at least two points to the right from s. At the beginning of the second phase, the player A selects any two adjacent subintervals and eliminates all other subintervals. In general, at the second phase, B selects points and A selects intervals. More specifically, player B on his move selects p distinct points on the interval and player A on her move selects any two adjacent subintervals and eliminates all other subintervals. The goal of the game is the same as in the single-processor case. It is obvious that on the first stage of the game player B must select all points in an increasing order from one p-probe to another.

Remark 3.2. At first, we will describe the optimal unbounded searching algorithm with one processing element, (PE, for short). Subsequently, we will describe and discuss the parallel minimax unbounded search with p PEs. As it will be demonstrated, the case where p is an *even* integer is simpler than the case where p is *odd*.

4. Structure of Unbounded Sequential Search

Consider a finite interval K, *i.e.*, its length $|K| < \infty$. In the case, if it is known a priori that $s \in K$, then we can t-detect maximizer s for at most $\left(\log_\alpha |K|/t\right)\left[1 + o\left(|K|/t\right)\right]$ probes, where

$$\alpha = \left(1+\sqrt{5}\right)/2, \text{ [16,17]}. \tag{4.1}$$

However, the situation is more complicated if f is defined on unbounded interval $I = (0, \infty)$. In this case we divide entire interval I into an infinite set of finite subintervals of uncertainty $I_1 := (0, q_1]$; $I_2 := (q_1, q_2]$; $\cdots$; $I_k := (q_{k-1}, q_k]$; $\cdots$ where

$$I = \bigcup_{k=1}^{\infty} I_k. \tag{4.2}$$

In this paper it is demonstrated that a minimax search algorithm consists of two major modes: a *scanning* (expanding) mode and a *detecting* (contracting) mode. Let us assume for simplicity of notations that for all integer k

$f_k := f(q_k)$, *i.e.*, f_k is a result of the *k*-th probe.

Definition 4.1. A search algorithm is in the *scanning mode* while for all $q_1 < q_2 < \cdots < q_k$ function *f* satisfies inequalities $f_1 < f_2 < \cdots < f_k$.

In the scanning mode we probe intervals $I_1, I_2, \cdots, I_k, \cdots$ until this mode terminates.

Definition 4.2. We say that a search algorithm is in *l-th state* $p_l = \{r_{l-1}, r_l\}$ of the scanning mode if *l*-th interval of uncertainty $I_l = (q_{l-1}, q_l]$ is to be eliminated and q_{l+1} is the next evaluation point. Here $r_{l-1} := q_l - q_{l-1} = |I_l|$; $r_l = q_{l+1} - q_l = |I_{l+1}|$.

Remark 4.1. If $f_l \ge f_{l+1}$, then the search switches into the detecting mode with initial state $\{r_{l-1}, r_l\}$, [16]. However, if $f_l < f_{l+1}$, then the search moves to the next p_{l+1} state of the scanning mode. As a result, the interval of uncertainty I_{l+1} is eliminated (since $s \notin I_{l+1}$) and the interval I_{l+2} is the next to be examined. Since at any state the search can switch into the detecting mode, the dilemma is whether to select interval I_{l+2} as small as possible (in preparation for this switch) or, if the search continues to stay in the scanning mode, to select I_{l+2} as large as possible. The dilemma indicates that there must be an optimal choice of I_{l+2}.

In the detecting mode, we can use an optimal strategy, [10,16,26], which locates *s* on *t*-interval. To design a minimax search algorithm, we must select all $q_1 < q_2 < \cdots < q_{k+1}$ in such a way, that the total number of required probes on both modes is minimal in the worst case.

Definition 4.3. We say that a set of points (q_i, q_j, q_k) is a *detecting triplet* if

$$f_i < f_j \ge f_k, \text{ where } q_i < q_j < q_k. \tag{4.3}$$

If (q_i, q_j, q_k) is a detecting triplet and *f* is a unimodal function, then maximizer *s* satisfies inequality $q_i < s \le q_k$, [17].

Definition 4.4. In the following consideration, $U(b,c)$ means the minimal total number of required probes for both modes in order to detect maximizer *s* in the worst case if $s \in (b,c]$.

In the following discussion, we assume that *t* = 1, unless it is specified otherwise.

Proposition 4.1. If *f* is an unimodal function and $s \in (F_{m-1} - 1, F_m - 1]$, but this is a priori unknown, then for all $m \ge 3$ *s* is detectable after $2(m-2)$ probes in the worst case, *i.e.*,

$$U\left(F_{m-1} - 1, F_m - 1\right) = 2\left(m-2\right) \tag{4.4}$$

where $m-1$ probes are used in the scanning mode and $m-3$ probes in the detecting mode. Here F_m is *m*-th Fibonacci number: $F_1 = F_2 = 1$; $F_m = F_{m-2} + F_{m-1}$ for $m > 2$.

Proof {by induction}: We will demonstrate that in the scanning mode the optimal evaluation points $q_1^o, q_2^o, \cdots, q_{k-1}^o, q_k^o$ must satisfy these properties: $q_1^o := 1$, $q_k^o - q_{k-1}^o := F_k$, for all $k \ge 2$, *i.e.*,

$$q_k^o := \sum_{i=1}^{k} F_i = F_{k+2} - 2. \tag{4.5}$$

Remark 4.3. First, we demonstrate how to find an approximation *a* of *s* that satisfies inequality $|s-a| \le t$, *i.e.*, $a - t \le s \le a + t$. Then we will show how to adjust the algorithm, if $s \in (n-1, n]$.

1). Let $s \in (F_3 - 2, F_4 - 2] = (0,1]$. Consider $1 \le q_1 < q_2 \le 2$. Then $f_1 > f_2$. Hence, Proposition 4.1 implies that $s \in (0, q_2)$. Thus, $a = q_2/2$ implies that $a \in (0,1]$; therefore $|s-a| \le 1 = t$. It is easy to verify that, if $s \in (0, 1+\delta)$ and $\delta > 0$, then, in the worst case, two probes are not sufficient for detection of *s* on *t*-interval. Indeed, the adversary can select such *s* and *f* that satisfy the following inequalities: $1 < q_1 < s < q_2$ and $f_1 < f_2$. Hence, in that case, *s* is not *t*-detectable on $(0,1]$ interval after two probes.

2) Let now $s \in (F_m - 2, F_{m+1} - 2)$ and $m \le k$. Then *s* can be *t*-detected for $\bigcup_s (F_m - 2, F_{m+1} - 2) = 2m - 1$ probes. If $m = k$, then *k* probes were used in the scanning mode and $k-3$ probes in the detecting mode.

3) Let $s \in (F_{k+1} - 1, F_{k+2} - 1]$, but it is a priori unknown. Let for all $i = 1, 2, \cdots, k+1, k+2$ the probes are taken in the points $q_i := F_{i+2} - 1$.

In this case the following inequalities hold $f_1 < f_2 < \cdots < f_k < f_{k+1} \ge f_{k+2}$.

Since $f_k < f_{k+1} \ge f_{k+2}$, then (q_k, q_{k+1}, q_{k+2}) is a detecting triplet, and, as a result, the search is in the detecting state $\{F_{k+1}, F_{k+2}\}$. Then from [8,10,26], using the optimal search algorithm, we can detect *s* with accuracy *t* = 1 for additional *k* evaluations of *f*. Hence the minimal total number of required probes for both modes is equal $U_t(F_{k+1} - 2, F_{k+2} - 2) = 2k - 1$. Q.E.T.

5. Optimal Balanced Sequential Search

5.1. The Algorithm

```
Assign a required accuracy t; {the scanning mode of the algorithm begins};
                    $L := t$;  $R := 2t$;
while  $f(L) < f(R)$  do begin
        $temp := L$;  $L := R$;  $R := R + temp + t$;        (5.1)
{(5.1) generates a sequence of probing states $\{F_1, F_2\}, \cdots, \{F_{k-1}, F_k\}$ }; end;
{the maximizer s is detected: $s \in (temp, R)$; the algorithm is in the detecting state
{ $L - temp$ , $R - L$ };
```

{the following steps describe the optimal detecting algorithm-see [16]};

```
assign
        $A := temp;\quad B := R;\quad R := A + B - L;$        (5.2)
repeat
if $f(L) > f(R)$ then
        $temp := L; L := 2R - B; B := R;$
        $R := temp; \{s \in (A, R)\};$        (5.3)
else
        $temp := R; R := 2L - A; A := L;$
        $L := temp; \{s \in (L, B)\};$        (5.4)
{(5.3) and/or (5.4) generate a sequence of detecting states $\{F_{k-1}, F_k\}, \cdots, \{F_1, F_2\}$ };
until $(B - A)/2 \le t$;
assign $a := (A + B)/2$; {$a$ is the approximation of the maximizer: $|s - a| \le t$ }; stop.
```

The algorithm described above is called V-algorithm.

5.2. Optimality of Sequential Search

Proposition 5.1. The number of required probes for *t*-detection of a maximizer described in Proposition 3.1 is minimal in the worst case.

Proof. The algorithm consists of the scanning and detecting modes. In the scanning mode (SM) the search is sequentially in the probing states $p_1, p_2, \cdots, p_m$ where

$$p_1 := \{F_1, F_2\}, \cdots, p_m := \{F_m, F_{m+1}\}. \qquad (5.5)$$

On the other hand, in the detecting mode (DM) the algorithm is in the detecting states $\{F_m, F_{m+1}\}$, $\cdots$, $\{F_1, F_2\}$. It is known that all detecting states $\{F_m, F_{m+1}\}$, $\cdots$, $\{F_1, F_2\}$ are optimal (there are no other strategies that can *t*-detect *s* for a smaller number of probes). At the same time the entire SM is a mirror image of the DM. Indeed, from the beginning to the end of the SM the search goes from scanning state $\{F_1, F_2\}$ to scanning state $\{F_m, F_{m+1}\}$, while in the DM the search goes from detecting state $\{F_m, F_{m+1}\}$ to detecting state $\{F_1, F_2\}$. Thus, both modes (scanning and detecting) are optimal; therefore, the entire algorithm is optimal.

6. Complexity of Minimax Sequential Search

Let us compare the optimal search algorithms for two cases:

1) Maximizer $s \in (b, F_m - 1]$, but this is a priori unknown; here *b* is a positive integer;

2) It is known a priori that $s \in (0, F_m)$. Let $B(b, c)$ be the minimal total number of required probes for *t*-detection of *s* in the worst case if $s \in (b, c)$.

From Proposition 4.1, if $b = F_{m-1} - 1$ and $m \ge 2$, then

$$U(F_{m-1} - 1, F_m - 1) = 2(m - 2). \qquad (6.1)$$

However, if $b = 0$, then the following inequality holds:

$$U(0, F_m - 1) \le 2(m - 2). \qquad (6.2)$$

From [11] it follows that, if $m \ge 4$, then $B(0, F_m) = m - 2$, otherwise

$$B(0, F_m) = 0. \qquad (6.3)$$

(6.1) and (6.3) imply that for all $m \ge 4$

$$U(0, F_m - 1) \le 2B(0, F_m) = 2(m - 2). \qquad (6.4)$$

In general, if $s \in (a, b] \subset (F_{m-1} - 1, F_m - 1]$, but this is a priori unknown, then

$$U(a, b) = 2\left\lceil \log_\alpha \left(b\sqrt{5}\right) \right\rceil \left[1 + o(b)\right] = \Theta(\log b) \qquad (6.5)$$

where α is defined in (4.1).

Equality (6.5) follows from the fact that

$$F_m = \left(\alpha^m - \sigma^m\right)/\sqrt{5}, \ [13,14]. \qquad (6.6)$$

where $\sigma = \left|\left(1 - \sqrt{5}\right)/2\right| < 1$, therefore $\lim \sigma^m = 0$ if $m \to \infty$.

Thus,

$$F_m = \left(\alpha^m/\sqrt{5}\right)\left[1 + o(\alpha)\right]. \qquad (6.7)$$

If $F_{m-1} \le P \le F_m - 1$, then

$$U(F_{m-1} - 1, P) = 2(m - 2). \qquad (6.8)$$

The complexities (6.1) and (6.5) can be further reduced if any prior information is available, [6,17], or if a searching algorithm is based on a randomized approach, [19,30].

Proposition 6.1. Let $c(n)$ be the minimal number in the worst case of the required probes to detect *s* on a priori unknown interval $s \in (n - 1, n]$. If

$$F_{m-1} \le 2n \le F_m - 1, \qquad (6.9)$$

then

$$c(n) = 2(m - 2) \qquad (6.10)$$

Proof. First of all, the relations (6.1), (6.2), (6.5) and (6.8) are based in the previously made assumption that *t* := 1. From this assumption it follows that maximizer is detectable on an interval of length two, $\{a - 1 \le s \le a + 1\}$. In order to find the complexity of the algorithm if $s \in (n - 1, n]$, the scale of the search must be decreased twice, *i.e.*, we must select *t* := 1/2. Two cases must be considered:

Case 1: If

$$\left(F_{m-1}-1\right)t < n-1 \text{ and } n \le \left(F_m -1\right)t\ ; \quad (6.11)$$

then (6.10) is implied by (6.5) if t :=1/2.

Case 2: If

$$\left(F_{m-1}\right)t \le n \le \left(F_m -1\right)t \text{ and } \left(F_{m-1}-1\right)t > n-1, \quad (6.12)$$

i.e., the case where $\left(F_{m-1}-1\right)t$ is in the middle of the interval $\left(n-1,n\right]$.

It occurs if $m \bmod 3 = 0.$ In this case the left half of the interval $\left(n-1,n\right]$ is out of interval $\left[\left(F_{m-1}-1\right)t, \left(F_m -1\right)t\right]$, *i.e.*, $\left(n-1,n-1/2\right] \not\subset \left(\left(F_{m-1}-1\right)t, \left(F_m -1\right)t\right]$ and, as a result, fewer probes are required for t-detection of s. However, in the worst case, the maximizer may be on the right half of interval $\left(n-1,n\right]$, hence $c(n) = 2(m-2)$ for both cases. For illustration see **Table 6.1** below.

Thus, (6.8) implies that

$$c\left(n\right) = 2\left\lceil \log_{\alpha}\left(2n\sqrt{5}\right)\right\rceil\left[1+o\left(n\right)\right]-4. \quad (6.13)$$

7. Estimated Interval of Uncertainty

In many applications, an upper bound value Q on maximizer s can be estimated from a feasibility study. Let $T = \alpha\sqrt{Q/\sqrt{5}} = d\sqrt{Q}.$ Here $d = \alpha/\sqrt[4]{5} = 1.082.$

Proposition 7.1. If $s \le T/\alpha$, then V-algorithm requires fewer probes than Kiefer's algorithm, [14]; if $T/\alpha < s \le T$, then Kiefer's algorithm and V-algorithm require the same number of probes; if $T < s \le Q$, then Kiefer's algorithm requires fewer probes than V-algorithm.

Proof. Let $T := F_m - 1$ and $Q := F_{2m-2}$. Then in the worst case

$$\begin{aligned} &U\left(0,F_{m-1}-1\right) < U\left(F_{m-1}-1,F_m -1\right) \\ &= B\left(0,F_{2m-2}\right) = 2\left(m-2\right) \end{aligned} . \quad (7.1)$$

It is easy to check that the proof follows from (6.7) and from the fact that

$$F_{2m-2} = \left(F_m/\alpha\right)^2\sqrt{5}\left[1+o\left(m\right)\right]. \quad (7.2)$$

Preliminary results on analysis of the optimal algorithm searching for the maximum of a multimodal function on the infinite interval are provided in [28].

Table 6.1. Total number of probes as function of *n*.

$4 \le n \le 6$	$494 \le n \le 798$	$60{,}697 \le n \le 98{,}208$
$c\left(n\right)=10$	$c\left(n\right)=30$	$c\left(n\right)=50$

8. Parallel Search: Basic Properties

If several processors are available, then, as it is indicated in [29], the algorithm can be executed in a parallel mode. [13,15] are the earliest papers on a parallel search for a maximum of a unimodal function of one variable on a finite interval, that are known to the author of this paper. Although the optimal search strategies in both papers are in essence identical, the formulation of the problem is different. The proof of optimality of the search is more detailed in [15]. [2] provides an idea of a parallel algorithm searching for a maximum of a unimodal function on a unbounded interval. This idea is based on an application of the Kraft's inequality formalism, is provided in [2]. The authors indicate that the approach they used to construct an infinite series of near-optimal algorithms for the unbounded search with a single processor can be expanded for a multiprocessor case. However, no details are provided.

The search algorithm described in this paper is based on the following properties.

Proposition 8.1: Let us consider p arbitrary points $q_1,\cdots,q_p$ that satisfy inequalities

$$0 < q_1 < \cdots < q_p < \infty\ . \quad (8.1)$$

If

$$f_1 < f_2 < \cdots < f_{p-1} < f_p\ , \quad (8.2)$$

then maximizer s is greater than q_{p-1}; if

$$f_1 > f_2 > \cdots > f_{p-1} > f_p\ , \quad (8.3)$$

then s is smaller than q_2; if

$$f_1 < \cdots < f_{j-1} < f_j \ge f_{j+1} > \cdots > f_p, \quad (8.4)$$

then s is greater than q_{j-1} and does not exceed q_{j+1}, *i.e.*,

$$s \in \left(q_{j-1}, q_{j+1}\right]. \quad (8.5)$$

Proof follows immediately from unimodality of function *f*.

9. Search on Finite Interval: Principle of Optimality

Definition 9.1. $I_m^p(u,v)$ is a minimal in the worst case interval of uncertainty containing maximizer s that can be detected after m p-probes if the search starts from the detecting state $\{u,v\}$.

9.1. Properties of $I_m^p\left(u,v\right)$

1) $I_m^p\left(u,v\right) > I_l^p\left(u,v\right)$ if $m < l$; (9.1)

(Effectiveness of p-probes);

2) $I_m^p(u_1,v_1) > I_l^p(u_2,v_2)$ if $u_1 \ge u_2$; $v_1 \ge v_2$; (9.2)
(Monotonicity of uncertainty);

3) $I_m^p(u,v) = I_m^p(v,u)$; (Symmetricity); (9.3)

4) $I_m^p(u,v) > I_m^q(u,v)$ if $p < q$; (9.4)
(Efficiency of parallelization);

5) $I_m^p(cu,cv) = cI_m^p(u,v)$ for every $c > 0$; (9.5)
(Homogeneity).

9.2. Properties of $I_m^p(u,v)$ if $p = 2$

Proposition 9.1: Let $u \ge v$. Then

$$I_m^2(u,v) = \min\begin{cases}\min\limits_{y_1<u,y_3<v} \max\left\{I_{m-1}^2\left[u-y_1,\max(y_1,y_3)\right], I_{m-1}^2\left[y_3,\max(u-y_1,v-y_3)\right]\right\};\\ \min\limits_{y_1+y_2<u} \max\left\{I_{m-1}^2\left[y_2,\max(y_1,u-y_1-y_2)\right], I_{m-1}^2\left[u-y_1-y_2,\max(y_2,v)\right]\right\}.\end{cases} \quad (9.6)$$

(9.6) is a functional equation of dynamic programming; it implies the following property. Proofs of this and the following Propositions 9.2-9.5 are provided in the Appendix.

Proposition 9.2: Let $u \ge v$; then

$$I_m^2(u,v) = \begin{cases} I_{m-1}^2\left(v,\dfrac{u-v}{2}\right) \text{ if } u > v;\\ I_{m-1}^2(u-z,z);\ 0<z<u \text{ if } u = v.\end{cases} \quad (9.7)$$

9.3. Odd Number of Processors

Proposition 9.3: If $p = 2r-1$; and $u \ge v$, then

$$I_m^{2r-1}(u,v) = \begin{cases} I_{m-1}^{2r-1}\left[(r-k+1)v-ku, ku-(r-k)v\right]/k,\ k/(r-k+1) < v/u \le k/(r-k);\\ I_{m-1}^{2r-1}\left[(r-k)v-ku,(k+1)u-(r-k)v\right]/(r-k),\ k < (r-k)v/u \le (k+1);\end{cases} \quad (9.8)$$

where for $1 \le k \le \lfloor r/2 \rfloor$ $(r-k+1)v-ku \ge ku-(r-k)v$ in the upper branch of (9.8) and for $0 \le k \le \lfloor r/2 \rfloor$ $(r-k)v-ku \ge (k+1)u-(r-k)v$ in the lower branch of (9.8).

9.4. Even Number of Processors

Proposition 9.4: If $p = 2r$ and $u \ge v$, then for all $1 \le k \le \lfloor r/2 \rfloor$ the following dynamic programming equation holds:

$$I_m^{2r}(u,v) = \begin{cases} I_{m-1}^{2r}\left[(r-k+1)v-ku,(k+1)u-(r-k)v\right]/(r+1),\ k/(r-k+1) < v/u \le (k+1)/(r-k);\\ I_{m-1}^{2r}\left[(u+v)/(r+1)-z,z\right],\ ku=(r-k+1)v,\ 0<z<(u+v)/(k+1).\end{cases} \quad (9.9)$$

9.5. Defining Rules of Optimal Detecting States

Definition 9.2. If there exists a pair of positive numbers c and d such that $c>d$ and for all non-negative numbers u and v

$$u+v \le c+d \text{ and } I_m^p(u,v) = I_m^p(c,d), \quad (9.10)$$

then $\{c,d\}$ is an *optimal* detecting state.

Definition 9.3. Let $\{c_k,d_k\}$ be the optimal detecting state starting from which s can be located after k additional p-probes.

The following two propositions can be proved by induction:

Proposition 9.5. if p is *odd*, then

$$I_m^{2r-1}(c_m,d_m) = I_{m-1}^{2r-1}(d_m,c_m/r-d_m), \quad (9.11)$$

which means that

$$c_{m-1} := d_m;\quad d_{m-1} := c_m/r-d_m; \quad (9.12)$$

Proposition 9.6. if p is *even*, then

$$I_m^{2r}(c_m,d_m) = I_{m-1}^{2r}\left[(c_m+d_m)/(r+1)-d_m,d_m\right],$$

which means that

$$d_{m-1} := d_m = z;\ c_{m-1} := (c_m+d_m)/(r+1)-d_{m-1}. \quad (9.13)$$

Remark 9.1. $d_m = const$ in (9.13).

1) and 2) imply *defining rules* (9.12) and (9.13) for optimal detecting states $\{c_k,d_k\}$; here $0<z<1$;

$$t := 1;\ d_0 := z;\ c_0 := t - z;\ r := \lceil p/2 \rceil. \qquad (9.14)$$

Proposition (1) and (2) imply the *defining rules* for optimal detecting states $\{c_k, d_k\}$: if p is odd, then for all $k \ge 1$

$$c_k := r\left(c_{k-1} + d_{k-1}\right);\ d_k := c_{k-1}; \qquad (9.15)$$

if p is even, then for all $k \ge 0$

$$\begin{aligned} d_k &:= z; \\ c_k &:= (r+1)\left(c_{k-1} + d_{k-1}\right) - d_{k-1} \qquad (9.16) \\ &= (r+1)c_{k-1} + r d_{k-1} = t(r+1)^k - z. \end{aligned}$$

Both these rules for p odd and even can be generalized as: assign

$$\phi := (p+1) \bmod 2; \qquad (9.17)$$

then

$$\begin{aligned} c_k &:= \lceil (p+1)/2 \rceil c_{k-1} - \phi d_{k-1}; \\ d_k &:= (p \bmod 2) c_{k-1} + \phi d_{k-1}. \end{aligned} \qquad (9.18)$$

The following two examples and **Table 9.1**

$$\{\text{with } t = 1;\ z = 1/2;\ c_0 = 1/2;\ d_0 = 1/2\} \qquad (9.19)$$

show how optimal search steps c_k and d_k are changing for various number of processors.

Example 9.1. Let $p = 3$; then $r := \lceil p/2 \rceil = 2$;
$c_1 := r(1/2 + 1/2) = 2$; $d_1 := 1/2$;
$c_2 := r(c_1 + d_1) = 2(2 + 1/2) = 5$; $d_2 := c_1 = 2$;
$c_3 := r(c_2 + d_2) = 2(5 + 2) = 14$; $d_3 := c_2 = 5$; $\cdots$

Example 9.2: Let $p = 4$; then for all $k \ge 1$ $d_k := 1/2$; and

$$c_1 := 3(c_0 + d_0) - d_0 = 3 - 1/2;$$

$$c_2 := 3(c_1 + d_1) - d_1 = 3^2 - 1/2;$$

$$c_3 := 3(c_2 + d_2) - d_2 = 3^3 - 1/2;\ \cdots$$

10. Search on Infinite Interval with Even Number of Processors $\{p = 2r\}$

10.1. Scanning Mode

Let us select the first p probes $q_{11}, q_{21}, \cdots, q_{p1}$.

If maximizer $s \in \left(0, q_{p-1,1}\right]$ and $q_{i1} - q_{i-2,1} \le 1$ for all $2 \le i \le p$, then s will be detected on a finite interval after the very first p-probe.

In general, if $s \in \left(q_{1,k}, q_{p-1,k}\right]$, then s will be detected on a finite interval after k p-probes.

10.2. Detecting Mode

Suppose the search is in the $(k+1)$-th detecting state. It means that at most k + 1 p-probe are required for t-detection of the maximizer s. p-probes will divide the larger interval c_{k+1} into p + 1 alternating sub-intervals $d_k, c_k, d_k, c_k, \cdots, c_k, d_k$. In the k-th detecting state, the search will be either in $\{d_k, c_k\}$ state or in the $\{c_k, d_k\}$ state. Both of these states are equivalent, *i.e.*, they require the same number of p-probes for the t-detection and the symmetrical choice of probes. Schematically it can be described in the following diagram:

$$\left\{d_{k+1}, c_{k+1}\right\} \to \left[c_k; d_k, c_k, d_k, c_k, \cdots, d_k, c_k, d_k\right] \to \left\{d_k, c_k\right\} \qquad (10.1)$$

Here a semicolon separates the "leftover" interval d_{k+1} of the previous state from the new sub-intervals $d_k, c_k, d_k, c_k, \cdots, c_k, d_k$, where the pair (d_k, c_k) is repeated r times.

It is important to notice for further application that the detecting states described above are not unique. Indeed, let us consider the search with p = 6 and let the search be in the detecting state $\{x, 8 - x\}$, where x is an integer variable, and $1 \le x \le 7$. Then for any x, the maximizer s is t-detectable after one p-probe only: divide the left interval into x equal subintervals using $x - 1$ probes and divide the right interval into $8 - x$ equal intervals using $8 - x - 1$ probes. Let x = 5. Applying the schematic description (10.1) of search we have
$\{5, 3\} \to [1,1,1,1,1;1,1,1] \to \{1,1\}$. If $x = 1$, then
$\{1, 7\} \to [1;1,1,1,1,1,1,1] \to \{1,1\}$. In general, for any even p we consider a detecting state $\{1, 2(r+1)^k - 1\}$. Let us divide the right interval onto p subintervals with alternating lengths equal $2(r+1)^{k-1} - 1$ and *one*. Then from the diagram (10.1) one can see that

$$\begin{aligned} &\left\{1, (r+1)^k - 1\right\} \\ &\to \left[1; 2(r+1)^{k-1} - 1, 1, 2(r+1)^{k-1} - 1, 1, \cdots, 1, 2(r+1)^{k-1} - 1\right] \\ &\to \left\{1, 2(r+1)^{k-1} - 1\right\} \to \left\{1, 2(r+1) - 1\right\} \\ &= \{1, p+1\} \to \left[1; 1, 1, \cdots, 1, 1\right] \to \{1, 1\}. \end{aligned}$$

Therefore, the detecting state $\{1, 2(r+1)^k - 1\}$ is the optimal detecting state starting from which s can be t-detected after k p-probes.

Table 9.1. Search intervals as functions of number of processors *p*.

p	2	3	4
c_1, d_1, z_1	1.5; 0.5; 2	2; 0.5; 2.5	2.5; 0.5; 3
$c_2; d_2; z_2$	3.5; 0.5; 4	5; 2; 7	8.5; 0.5; 9
$c_3; d_3; z_3$	7.5; 0.5; 8	14; 5; 19	26.5; 0.5; 27
p	5	6	7
c_1, d_1, z_1	3; 0.5; 3.5	3.5; 0.5; 4	4; 0.5; 4.5
$c_2; d_2; z_2$	10.5; 3; 13.5	15.5; 0.5; 16	18; 4; 22
$c_3; d_3; z_3$	40.5; 10.5; 51	63.5; 0.5; 64	88; 18; 106

11. Inter-Processor Communication Network

1) All PEs are connected with a linear bus;

2) Two adjacent PEs share a memory unit (MEM, for short), *i.e.*, PE(i) and PE(i + 1) are connected with MEM(i) and can read from it concurrently.

12. Speed-up and Efficiency of Parallelization

Let $b_{m(p)-1} < n \le b_{m(p)}$ and $b_{m(1)-1} < n \le b_{m(1)}$. (12.1)

Then for large $m(p)$ and $m(1)$ holds that

$$m(p)\log u(p) = m(1)\log u(1). \quad (12.2)$$

On the other hand, the numbers of required probes in parallel and sequential modes are respectively equal

$$c_p(n) = 2m(p)-1 \text{ and } c_1(n) = 2m(1)-1. \quad (12.3)$$

Let's define the speed-up of parallelization as

$$s_p(n) = c_1(n)/c_p(n). \quad (12.4)$$

Then (12.2) and (12.3) imply that

$$s_p(n) = \frac{\log u(p)}{\log u(1)}. \quad (12.5)$$

If efficiency of parallelization is defined as

$$e_p(n) := s_p(n)/p, \quad (12.6)$$

then

$$e_p(n) = c_1(n)/\left[pc_p(n)\right] = \frac{\log u(p)}{p\log u(1)}. \quad (12.7)$$

Since for the large number p of processors

$$u(p) \to \lceil p/2 \rceil + 1;$$

(see (A.15) and **Table A.1** in Appendix) (12.8)

therefore for a large n

$$s_p(n) = \frac{\log(p/2+1)}{\log\left[\left(1+\sqrt{5}\right)/2\right]} = \Theta(\log p); \quad (12.9)$$

and finally

$$e_p(n) = \Theta\left[(\log p)/p\right]. \quad (12.10)$$

13. Acknowledgements

I express my appreciation to Willard Miranker and Shmuel Winograd of Thomas J. Watson Research Center and Henryk Wozniakowski of Columbia University for their comments and discussions. I also thank my former students Swetha Medicherla and Aleksey Koval for suggestions that improved the style of this paper.

14. References

[1] J. L. Bentley and A. C.-C. Yao, "An Almost Optimal Algorithm for Unbounded Searching," *Information Processing Letters*, Vol. 5, No. 1, 1976, pp. 82-87. doi:10.1016/0020-0190(76)90071-5

[2] R. Beigel, "Unbounded Searching Algorithm," *SIAM Journal of Computing*, Vol. 19, No. 3, 1990, pp. 522-537. doi:10.1137/0219035

[3] E. M. Reingold and X. Shen, "More Nearly-Optimal Algorithms for Unbounded Searching, Part I, the Finite Case," *SIAM Journal of Computing*, Vol. 20, No. 1, 1991, pp. 156-183. doi:10.1137/0220010

[4] E. M. Reingold and X. Shen, "More Nearly-Optimal Algorithms for Unbounded Searching, Part II, the Transfinite Case," *SIAM Journal of Computing*, Vol. 20, No. 1, 1991, pp. 184-208. doi:10.1137/0220011

[5] A. S. Goldstein and E. M. Reingold, "A Fibonacci-Kraft Inequality and Discrete Unimodal Search," *SIAM Journal of Computing*, Vol. 22, No. 4, 1993, pp. 751-777. doi:10.1137/0222049

[6] A. S. Nemirovsky and D. B. Yudin, "Problems Complexity and Method Efficiency in Optimization," Wiley-Interscience, New York, 1983.

[7] J. F. Traub and H. Wozniakowski, "A General Theory of Optimal Algorithms," Academic Press, San Diego, 1980.

[8] J. H. Beamer and D. J. Wilder, "Minimax Optimization of Unimodal Function by Variable Block Search," *Management Science*, Vol. 16, 1970, pp. 629-641.

[9] D. Chasan and S. Gal, "On the Optimality of the Exponential Function for Some Minimax Problems," *SIAM Journal of Applied Mathematics*, Vol. 30, No. 2, 1976, pp. 324-348. doi:10.1137/0130032

[10] J. C. Kiefer, "Sequential Minimax Search for a Maximum," *Proceedings of American Mathematical Society*, Vol. 4, No. 3, 1953, pp. 502-506. doi:10.1090/S0002-9939-1953-0055639-3

[11] L. T. Oliver and D. J. Wilde, "Symmetric Sequential Minimax Search for a Maximum," *Fibonacci Quarterly*, Vol. 2, No. 3, 1964, pp. 169-175.

[12] C. Witzgall, "Fibonacci Search with Arbitrary First Evaluation," *Fibonacci Quaterly*, Vol. 10, No. 2, 1972, pp. 113-134.

[13] M. Avriel and D. J. Wilde, "Optimal Search for a Maximum with Sequences of Simultaneous Function Evaluations," *Management Science*, Vol. 12, No. 9, 1966, pp. 722-731. doi:10.1287/mnsc.12.9.722

[14] S. Gal and W. L. Miranker, "Optimal Sequential and Parallel Search for Finding a Root," *Journal of Combinatorial Theory, Series A*, Vol. 23, No. 1, 1977, pp. 1-14. doi:10.1016/0097-3165(77)90074-7

[15] R. Karp and W. L. Miranker, "Parallel Minimax Search for a Maximum," *Journal of Combinatorial Theory*, Vol. 4, 1968, pp. 59-90.

[16] B. Verkhovsky, "Optimal Search Algorithm for Extrema of a Discrete Periodic Bimodal Function," *Journal of Complexity*, Vol. 5, No. 2, 1989, pp. 238-250. doi:10.1016/0885-064X(89)90006-X

[17] B. Veroy (Verkhovsky), "Optimal Algorithm for Search of Extrema of a Bimodal Function," *Journal of Complexity*, Vol. 2, No. 4, 1986, pp. 323-332. doi:10.1016/0885-064X(86)90010-5

[18] B. Veroy, "Optimal Search Algorithm for a Minimum of a Discrete Periodic Bimodal Function," *Information Processing Letters*, Vol. 29, No. 5, 1988, pp. 233-239. doi:10.1016/0020-0190(88)90115-9

[19] S. Gal, "Sequential Minimax Search for a Maximum When Prior Information Is Available," *SIAM Journal of Applied Mathematics*, Vol. 21, No. 4, 1971, pp. 590-595. doi:10.1137/0121063

[20] Yu. I. Neymark and R. T. Strongin, "Informative Approach to a Problem of Search for an Extremum of a Function," *Technicheskaya Kibernetika*, Vol. 1, 1966, pp. 7-26.

[21] R. Horst and P. M. Pardalos, "Handbook of Global Optimization," Kluwer Academic Publishers, Norwell, 1994.

[22] B. O. Shubert, "A Sequential Method Seeking the Global Maximum of a Function," *SIAM Journal of Numerical Analysis*, Vol. 3, No. 1, 1972, pp. 43-51.

[23] L. N. Timonov, "Search Algorithm for a Global Extremum," *Technicheskaya Kibernetika*, Vol. 3, 1977, pp. 53-60.

[24] A. J. Hoffman and P. Wolfe, "Minimizing a Unimodal Function of Two Integer Variables," *Mathematical programming*, *II*. *Mathematical Programming Studies*, Vol. 25, 1985, pp. 76-87.

[25] A. I. Kuzovkin, "A Generalization of Fibonacci Search to the Multidimensional Case," *Èkonomka i Matematicheskie Metody*, Vol. 21, No. 4, 1968, pp. 931-940.

[26] J. C. Kiefer, "Optimal Sequential Search and Approximation Methods under Minimum Regularity Assumptions," *SIAM Journal of Applied Mathematics*, Vol. 5, 1957, pp. 105-136. doi:10.1137/0105009

[27] S. Gal, "A Discrete Search Game," *SIAM Journal of Applied Mathematics*, Vol. 27, No. 4, 1974, pp. 641-648. doi:10.1137/0127054

[28] B. Verkhovsky, "Optimal Algorithm for Search of a Maximum of *n*-Modal Function on Infinite Interval," In: D. Du and P. M. Pardalos, Eds., *Minimax and Applications*, Academic Publisher, Kluwer, 1997, pp. 245-261.

[29] B. Verkhovsky, "Parallel Minimax Unbounded Searching for a Maximum of a Unimodal Function," *Research Report*, CIS-95-03, New Jersey Institute of Technology, Newark, 1995, pp. 1-24.

Appendix

A1. Complexity Analysis

A1.1. Basic Parameters

$a_k =$ interval added before *k*-th p-probe is computed;
$g_k =$ smaller interval on the *k*-th scanning state of the search;
$h_k =$ larger interval on the *k*-th scanning state of the search;
$w_k =$ interval of uncertainty eliminated from the search after the *k*-th p-probe;
$t_k =$ total interval added before the *k*-th p-probe is performed;
$b_k =$ total interval eliminated from the search as a result of *k* p-probes.

A1.2. Basic Relations: {*odd p*}

$$a_k = \lceil p/2 \rceil h_k + \lfloor p/2 \rfloor g_k = rh_k + (r-1) g_k$$
$$= r(g_k + h_k) - g_k; \quad \text{(A.1)}$$
$$a_k := h_k - h_{k-2};$$

w_k and b_k satisfy the following recursive relations for all $k \ge 3$:

$$w_k = a_k - h_k + g_k;\ w_k := h_k - h_{k-1}; \quad \text{(A.2)}$$

$$t_k := (h_{k+1} - h_1)/r; \quad \text{(A.3)}$$

$$b_k = b_{k-1} + w_k;\ b_k := t_k - h_{k-1}. \quad \text{(A.4)}$$

A1.3. Basic Relations: {*even p*}

$$g_k := z;\ h_k := (r+1)^k - z;\ w_1 := r-2;$$
$$w_k := r(r+1)^{k-1} - z; \quad \text{(A.5)}$$

$$b_k := \sum_{i=1}^{k} w_i = \sum_{i=1}^{k-1} r(r+1)^i - z = (r+1)^k - z. \quad \text{(A.6)}$$

Let us consider for all $k \ge 1$ sequences h_k, v_k, δ_k with the following defining rules:

$$h_k := v_k - \delta_k z;\ h_k = r(h_{k-1} + h_{k-2}); \quad \text{(A.7)}$$

where

$$v_{-1} := 0;\ v_0 := 1;\ \delta_{-1} := -1;\ \delta_0 := 1 \text{ and } 0 < z < 1. \quad \text{(A.8)}$$

(A.7) implies that

$$v_k := r(v_{k-1} + v_{k-2});\ \delta_k := r(\delta_{k-1} + \delta_{k-2}). \quad \text{(A.9)}$$

It is easy to demonstrate by induction that for all

$$k \ge 1\ \ \delta_k = rv_{k-2}. \quad \text{(A.10)}$$

Therefore

$$h_k = v_k - rv_{k-2} z. \quad \text{(A.11)}$$

If $p = 1$, then $v_k = F_k$, where all F_k are the Fibonacci numbers.

Thus, all v_k can be computed using the following formula:

$$v_k = \beta u^k + \omega w^k; \quad \text{(A.12)}$$

where *u* and *w* are roots of the equation

$$x^2 - r(x-1) = 0: \quad \text{(A.13)}$$

and β and ω satisfy the equations:

$$\beta + \omega = v_0 = 1;\ \beta u + \omega w = v_1 = r. \quad \text{(A.14)}$$

From (A.13)

$$u = \left(r + \sqrt{r(r+4)}\right)/2 = r + 1 - \delta(p)$$
$$w = \left(r - \sqrt{r(r+4)}\right)/2;\ r = (p+1)/2; \quad \text{(A.15)}$$

{see **Table A.1** for values of *u*(*p*)}.

For every *p* $|x_1| > 1 > |x_2|;\ u = x_1;\ w = x_2$. Indeed, from Viete theorem

$$w = -r/u. \quad \text{(A.16)}$$

Since $u > r$, then $-1 < w < 0$. From (A.15) and (A.16)

$$v_k = \beta u^k + o(u). \quad \text{(A.17)}$$

From (A.14)

$$\beta = \left[r(r+4) + (r-2)\sqrt{r(r+4)}\right]/2r(r+4). \quad \text{(A.18)}$$

Therefore, $\lim_{p\to\infty} \beta(p) = 1/2;\ \ \lim_{p\to\infty} \delta(p) = 0;$ *i.e.*,

$$\lim_{p\to\infty} u(p)/(r+1) = 1. \quad \text{(A.19)}$$

The latter limit in (A.19) means that for a large number of processors $u(p) = (p+3)/2$.

Examples A1: **Table A.1** shows relationship between odd *p* and *u*(*p*).

A2. Maximal Intervals Analyzed after *m* Parallel Probes

Proposition A.1: If $s \in (v_{m-1} - 1, v_m - 1)$, then *m* p-probes are required in the worst case to detect *the maximizer* on a final interval.

Proof is implied by (A.14) and (A.17).

Proposition A.2: If *p* is odd, then

Table A.1. ***u*(*p*) as function of odd number of processors *p*.**

p	*p* = 1	*p* = 5	*p* = 9
u(*p*)	$(1+\sqrt{5})/2 = 1.618$	$(3+\sqrt{21})/2 = 3.791$	$(5+\sqrt{45})/2 = 5.854$

$$c_p(n) = \left\lceil 2\log_{u(p)}\left[(v_m - 1)/\beta(p)\right]\right\rceil - 1 = \left\lceil 2\log_{u(p)}\left[(n+1)/\beta(p)\right]\right\rceil - 1; \quad (A.20)$$

and, if p is even, then

$$c_p(n) = \left\lceil 2\log_{r+1}(v_m - 1)\right\rceil - 1 = \left\lceil 2\log_{r+1}(n+1)\right\rceil - 1. \quad (A.21)$$

Proof: If $s \in (n-1, n) \subset (v_{m-1} - 1, v_m - 1]$, then $c(n) = 2m - 1$. Then the proof follows from (A.14) and (A.17) respectively.

A3. Proof of Proposition A 9.1

Proof: There are only three ways to decompose the intervals u and v using two evaluations (semicolons indicate the previous points that separate u and v):

1) $\{u,v\} \to [y_1; y_2, y_3, y_4] \to [u; y_2, y_3, v - y_2 - y_3]$; where $y_1 = u$; $y_2 + y_3 + y_4 = v$; (A.21)

2) $\{u,v\} \to [y_1, y_2; y_3, y_4] \to [y_1, u - y_1; y_3, v - y_3]$; where $y_1 + y_2 = u$; $y_3 + y_4 = v$; (A.22)

3) $\{u,v\} \to [y_1, y_2, y_3; y_4] \to [y_1, y_2, u - y_1 - y_2; v]$. where $y_1 + y_2 + y_3 = u$; $y_4 = v$. (A.23)

The following recursive equation is derived from the decomposition (A.21)-(A.23):

$$I_m^2(u,v) = \min\begin{cases} \min\limits_{y_2,y_3} \max\left[I_{m-1}^2(u,y_2), I_{m-1}^2(y_2,y_3), I_{m-1}^2(y_3, v - y_2 - y_3)\right]; \\ \min\limits_{y_1,y_3} \max\left[I_{m-1}^2(y_1, u - y_1), I_{m-1}^2(u - y_1, y_3), I_{m-1}^2(y_3, v - y_3)\right]; \\ \min\limits_{y_1,y_2} \max\left[I_{m-1}^2(y_1,y_2), I_{m-1}^2(y_2, u - y_1 - y_2), I_{m-1}^2(u - y_1 - y_2, v)\right]. \end{cases} \quad (A.24)$$

Taking into account that $u + y_2 > v + y_2 > y_2 + y_3$ and that $u + y_2 > v > v - y_2$ we can eliminate the second and the third terms in the upper branch of the functional Equation (A.24).

On the other hand the first term of the upper branch in (A.24) can itself be eliminated since

$$\min_{y_2 < v} I_{m-1}^2(u, y_2) \geq \min_{y_1 < u} I_{m-1}^2(u - y_1, y_1). \quad (A.25)$$

In addition, $\min\limits_{0<y_1<u} I_{m-1}^2(y_1, u - y_1) \geq \min\limits_{0<y_3<v}(y_3, v - y_3)$.

Hence Equation (A.24) is reduced to Equation (9.1). Q.E.D.

A4. Proof of Proposition A9.2

$$I_m^2(u,v) = \begin{cases} I_{m-1}^2\left(v, \dfrac{u-v}{2}\right) \text{ if } u > v; \\ I_{m-1}^2(u - z, z);\ 0 < z < u \text{ if } u = v. \end{cases} \quad (A.26)$$

Proof: Since, in the worst case, the adversary can select two adjacent intervals with the largest sum, from optimality point of view, one must select the intervals in such a way that every two adjacent intervals have equal sums. It implies that the alternating intervals must have equal lengths: $y_1 = y_3 = w$; $y_2 = y_4 = z$.

Hence (A.26) implies that $y_1 = u - y_1 - y_2 = w$; $y_2 = v$.

Thus, $y_1 = (u - v)/2$ if $u > v$.

However, if $u = v$, then $y_1 = y_3 = z$; $u - y_1 = v - y_3 = u - z$.

Let us consider for every k = 1, $\cdots$, p + 1 a pair of equations:

$$\sum\nolimits_{i=1}^{k} y_i = u; \quad \sum\nolimits_{i=k+1}^{p+2} y_i = v; \quad (A.27)$$

where for all i = 2, $\cdots$, p + 2 $y_i > 0$.

A5. Proof of Proposition A9.3

$$I_m^p(u,v) = \min_{1\le k\le p+1} \max_{1\le i\le p+1} \min_{y_i, y_{i+1}} I_{m-1}^p(y_i, y_{i+1}). \quad (A.28)$$

Proof: There are p + 1 ways to represent the intervals u and v as sums in (A.27). The following dynamic programming equation describes recursive relations between the detecting states

$$I_m^p(u,v) = \min_{1\le k\le p+1} \min_{y1,\cdots,y_{p+2}} \max\left[I_{m-1}^p(y_1,y_2), I_{m-1}^p(y_2,y_3), \cdots, I_{m-1}^p(y_{k-1},y_k), I_{m-1}^p(y_k,y_{k+1}), \cdots, I_{m-1}^p(y_{p+1},y_{p+2})\right]$$

where all control variables $y_1, \cdots, y_{p+2}$ must satisfy constraints (A.27). Considering the worst case, a user *can* select such a function *f*, that the algorithm *must* select a pair of adjacent intervals with the largest sum.

A6. Proof of Proposition 9.4

If $p = 2r - 1$; and $u \geq v$, then

$$I_m^{2r-1}(u,v) = \begin{cases} I_{m-1}^{2r-1}\left[(r-k+1)v - ku, ku-(r-k)v\right]/k, & k/(r-k+1) < v/u \le k/(r-k); \\ I_{m-1}^{2r-1}\left[(r-k)v - ku, (k+1)u-(r-k)v\right]/(r-k), & k < (r-k)v/u \le (k+1); \end{cases} \quad \text{(A.29)}$$

Proof: Since an algorithm designer's goal is for a given number of p-probes to maximize an interval on which s can be located within a unit interval, it is obvious to select all the intervals in such a way that any two adjacent intervals would have the same sum. This search strategy means that intervals $y_1, y_2, y_3, \cdots, y_p, y_{p+1}, y_{p+2}$ must have alternate values: $y_1 = y_3 = w$; $y_2 = y_4 = z$; $\cdots$, *i.e.*, in general, for all $1 \le i \le p/2$ $y_{2i-1} = w$; $y_{2i} = z$.

A7. Proof of Proposition A9.5

If $p = 2r$ and $u \ge v$, then for all $1 \le k \le \lfloor r/2 \rfloor$ the following dynamic programming equations holds:

$$I_m^{2r}(u,v) = \begin{cases} I_{m-1}^{2r}\left[(r-k+1)v - ku, (k+1)u-(r-k)v\right]/(r+1), & k/(r-k+1) < v/u \le (k+1)/(r-k); \\ I_{m-1}^{2r}\left[(u+v)/(r+1) - z, z\right], & ku = (r-k+1)v,\ 0 < z < (u+v)/(k+1). \end{cases} \quad \text{(9.9)}$$

Proof is analogous to the proof of the Theorem 9.4.