Discussion Paper No. 619

RANDOM BINARY SEARCH: A RANDOMIZING ALGORITHM FOR GLOBAL OPTIMIZATION IN R^1[*]

by

Eitan Zemel[**]

August 1984

[*] Research for this paper was supported in part by National Science Foundation Grant No. ECS-8121941 to Northwestern University.

[**] Department of Managerial Economics and Decision Sciences, J. L. Kellogg Graduate School of Management, Northwestern University, Evanston, Illinois 60201 and the L. Recanati School of Business Administration, Tel Aviv University, Israel.

This paper was presented at the 5th Bonn Workshop on Combinatorial Optimization, June 1984.

ABSTRACT

We analyze the performance of randomized binary search, a randomizing algorithm for finding the global optimum of a multimodal one dimensional function. In particular, we give an effective (computable in linear time) formula for the sampling distribution, i.e., the probabilities of sampling the various local optima. We also obtain various interesting properties of this distribution which are relevant to the behavior of the algorithm in practice. Finally we discuss some issues related to adaptive search.

1. Introduction

There are numerous algorithms which can find the optimum of a unimodal function defined over a certain region. Yet, in practice, one often has to deal with functions which may possess more than one (local) optimum. One often approaches such problems by using an algorithm designed to handle the unimodal case. When applied to a multimodal function, such an algorithm typically yields a local optimum. The trick is to force the algorithm to produce new local optima as it is applied over and over again, until all local optima have been generated. This can be done deterministically, basically by enumeration, or stochastically, by introducing some random elements into the algorithm so that it follows a different computational path at each application. There is a large body of literature on the theory and practice of both methods. We mention here a representative sample of this research [1, 2, 3, 4, 6, 7, 8, 9, 10]. Two excellent treatments of this subject with exhaustive reference lists can be found in the recent theses of Timmer [10] and Boender [4].

In this note we consider a randomizing algorithm for global optimization. Such algorithms yield a random local optimum at each application. Thus, they can be viewed as sampling procedures, where a local optimum is sampled at each iteration from the set of local optima. A key issue in analyzing algorithms of this type is that of the sampling distribution, i.e., the probabilities of sampling the various local optima. Clearly, these probabilities are the key determinants of the algorithm's efficiency. Unfortunately, there are very few analytical results on the magnitude of these probabilities as the required analysis is typically very hard. However, as demonstrated below, such analysis is still possible for an extremely powerful and well-known algorithm.

The case analyzed in this note is the relatively simple but inherently basic and important one dimensional case. The algorithm we consider is an obvious randomizing version of the well-known and effective binary search. We present a detailed analysis of the (stochastic) performance of this algorithm and present an effective (computable in linear time) formula for the sampling distribution. We also examine several properties of this distribution which allow us to reach some interesting (and surprising) conclusions concerning this algorithm. Although our main motive in this study is global optimization, we have found the stochastic process which underlies the algorithm to be very interesting and elegant in itself.

The structure of this paper is the following. In section 2 we present the necessary preliminaries. The analysis is contained in section 3. Section 4 concerns the computation of the sampling probabilities. Their properties are discussed in section 5. Section 6 is concerned with the analysis of the completely symmetric case. Finally, in section 7, we explore some issues related to adaptive search.

2. Notations and Preliminaries

Let f be a continuous function of one variable and u = [a,b] a given interval over which f is defined. The function f may contain several local minima over u. We wish to find the global minimum, i.e., a point x^* ∈ u such that $f(x^*) < f(x)$ for all x ∈ u. In order to handle computational complexity issues for this task we make the following assumptions:

(1) f is almost everywhere differentiable. Moreover, for each point of differentiability, x, one can evaluate the function and its derivative in constant time (in fact, the sign of the derivative is all that is required).

(2) For simplicity we assume that the derivative of f is almost everywhere nonzero. (The reader will note that this assumption is not

required and is used here for economy of exposition only.)

(3) A constant δ is supplied in advance such that it is sufficient to identify a given point to within δ. (See [5] for a discussion of this issue.) This implies that any interval of size δ or less cannot contain more than one local optimum and that one can decide which among two local minima is better based on their specification to within such constant δ.

The randomizing algorithm we consider is a natural modification of binary search. At the k-th step, pick a random point z_k from the uniform distribution over the subinterval $u_k = [a_k,b_k]$ of $u \equiv u_1$ which is still under consideration and evaluate the sign of the derivative f at z_k. If that sign is positive then f contains a local minimum in $[a_k,z_k]$. We set $u_{k+1} = [a_k,z_k]$ and proceed to the k + 1 step. If the derivative is negative at z_k we proceed analogously. The iteration terminates when $|u_k|$, the length of u_k, is less than δ. When it does, a unique local minimum, $\bar{x}$, has been singled out. Obviously, the identity of this minimum is random. We can thus view the algorithm as a process of sampling a local minimum of f. We are interested in computing the sampling distribution. Before proceeding further the reader may find it instructive to examine the following three simple examples (figures 1-3) and establish on his own (at least qualitatively) the probabilities of the algorithm terminating at each of the local minima. We will examine the correct answers in section 5.

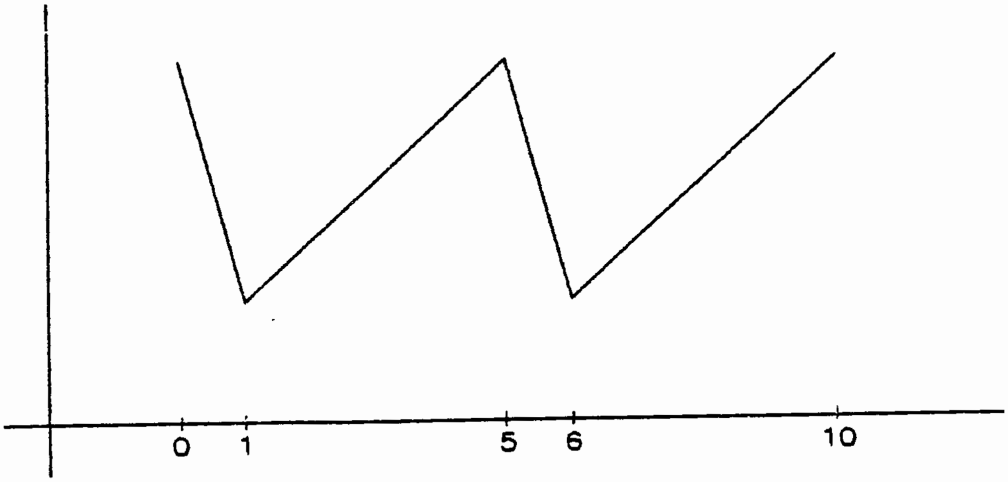

Figure 1

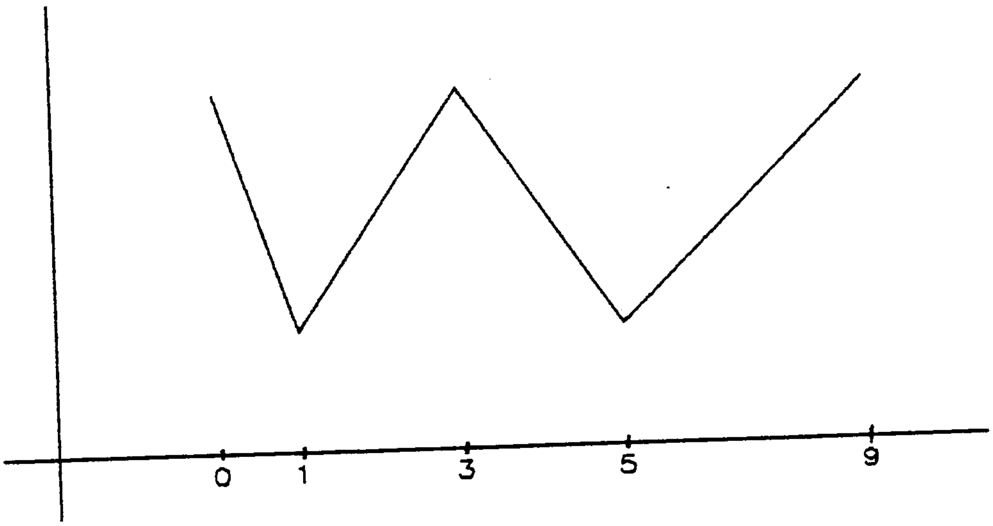

Figure 2

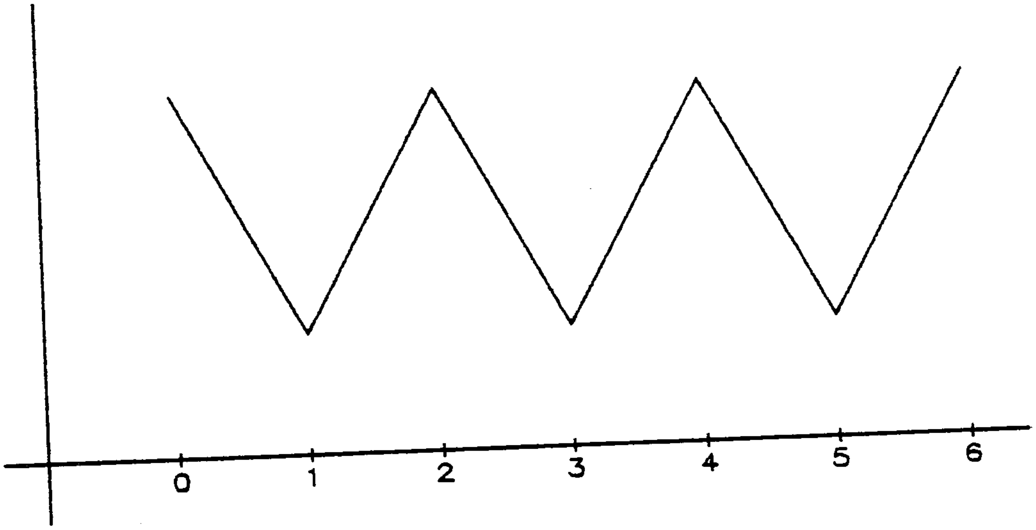

Figure 3

3. Analysis

Consider the interval u = [a,b] and denote the number of local minima in this interval by n (possibly including the end points of the interval). Let the positions of these minima be at $X = (x_1,x_2,...,x_n)$ with $a < x_1 < x_2,..., < x_n < b$. Then, the number of local maxima of f in u is n - 1, n, or n + 1, depending on whether one or two of the minima of f are obtained at end points. We use the following convention: if a or b is a local minimum of f then we include it also in the list of local maxima. Under this convention the number of local maxima is always n + 1. Denote their positions by $Y = (y_1, y_2, \ldots, y_{n+1})$ with $y_1 < y_2 < y_3, \ldots, < y_{n+1}$. Thus, $a = y_1 \leq x_1 < y_2, \ldots, < x_n \leq y_{n+1} = b$. Let $R_i = (y_i, x_i)$ and $L_i = (x_i, y_{i+1})$ $i = 1, \ldots, n$ be the open intervals surrounding the local minima x_i. We denote the lengths of these intervals by ℓ_i and r_i, respectively. The general case is depicted in figure 4. Figure 5 depicts the situation when a is a local minimum. This causes x_1 to coincide with y_1 and thus $L_1 = \emptyset$, $\ell_1 = 0$. For $i = 1, \ldots, n$ let $t_{2i-1} = \ell_i$, $t_{2i} = r_i$, $T = (t_1, \ldots, t_{2n})$. For each interval $v = [c,d] \subseteq u_1$ let $|v|$ denote its length.

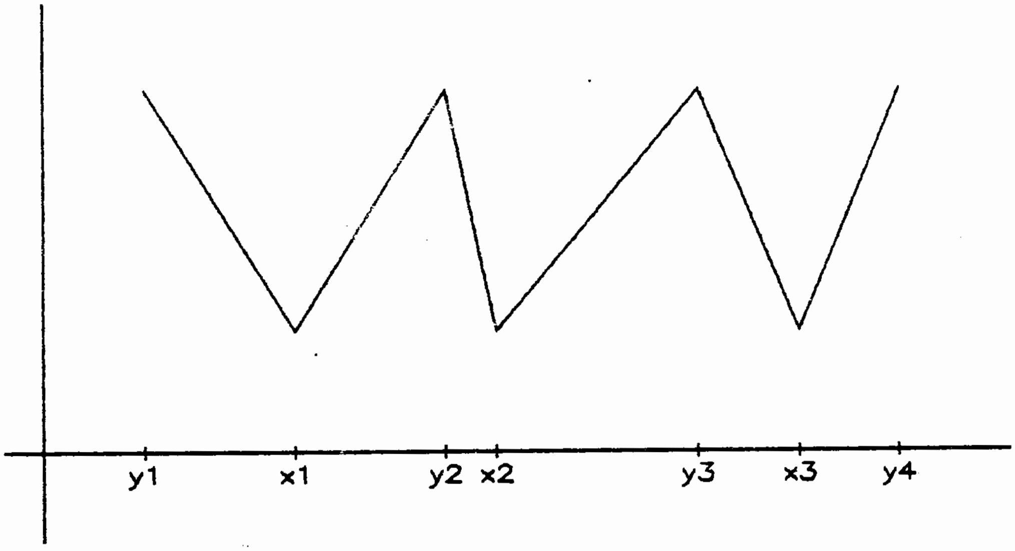

Figure 4

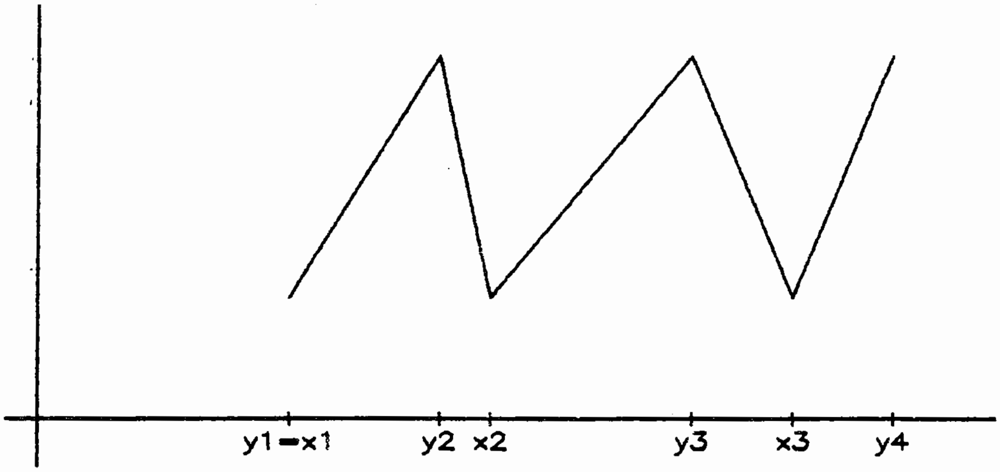

Figure 5

In general, the progress of the algorithm can be described by a (possibly infinite) nested sequence of intervals $U = (u_1 \supseteq u_2 \supseteq \ldots)$ and of points $Z = (z_1, z_2, \ldots)$ such that z_k is uniformly distributed over u_k and u_{k+1} is the intersection of u_k with one of the intervals $[a, z_k]$ or $[z_k, x]$ depending on the sign of the derivative of f at z_k. In principle, the sequence of points Z need not converge. However, this is highly unlikely. In fact, the probability of the interval shrinking in size to δ in a _finite_ number of steps can be made arbitrarily close to 1.

Theorem 1. Let $r = \log_3 (|u|/\delta)$ and let $k^* = 3mr$ for some constant $m > 1$. Let $t(m) = \frac{3}{8}(\frac{m-1}{m})^2$. Then

$$\Pr [|u_{k+1}| > \delta] < e^{-t(m)}$$

The proof of Theorem 1 is standard, and will be omitted (see, e.g., [11] or [12] for similar applications of the normal approximation to the binomial distribution). $\square$

The implications of Theorem 1 concern the computational complexity of each _iteration_. Note that assumption (1) implies that the work per _step_ is O(1). In principle, the work per iteration is a random variable which may assume arbitrarily large values. The theorem guarantees (probabilistically) that this random variable is also bounded by a constant, which depends, however, on $\log(|u|/\delta)$.

The only information used during the execution of a step is the sign of the derivative of f at the random points z_k. That sign is positive at points of $R \equiv \bigcup_{i=1}^{n} R_i$ and negative at $L \equiv \bigcup_{i=1}^{n} L_i$. Thus, the progress of the algorithm is completely specified by the positions and lengths of the intervals R_i, L_i,

$i = 1,\ldots,n$. This means that there is no loss of generality if we view f as a piecewise linear function with consecutive segments having derivatives of alternating signs, as depicted in figures 1-5. Moreover, since the position of the interval u on the x axis is obviously irrelevant, the probability sought depends only on the vector $T = (t_1,\ldots,t_{2n})$.

Denote the probability that the iteration terminates with the minimum x_i by $P_i(T)$ and let $P(T) = (p_1(T),p_2(T),\ldots,p_n(T))$ with $\sum_{i=1}^{n} p_i(T) = 1$. We now examine this probability in more detail.

Let $z \in u$ be arbitrary. Assume we split u at z obtaining two subintervals $ur(z) = [z,b]$ and $u\ell(z) = [a,z]$. Let the corresponding T vectors for each of these subintervals be $TR(z)$ and $TL(z)$, respectively. Splits play a crucial role in our analysis since each step is essentially a split at a random point z. In fact, we can easily state a recursive expression for $p_i(T)$ which is based on the observation that the probability of z falling in R_i or L_i is equal to $r_i/|u|$ and $\ell_i/|u|$, respectively, and given that z is in one of these intervals, it is distributed there uniformly. Thus, we have

<u>Theorem 2.</u>

$$(4) \qquad P_i(T) = \frac{1}{|u|}\Big[\Big(\sum_{j=1}^{i} \int_{y_j}^{x_j} p_{i-j+1}(TR(z))dz + \sum_{j=1}^{n} \int_{x_j}^{y_{j+1}} p_i(TL(z))dz\Big)\Big] \qquad \square$$

This looks like a hopeless formula. However, we will be able to simplify it considerably with the help of an extremely useful and interesting Theorem 3. To state this theorem, we now concentrate on a very special type of splits, namely, at the actual minima x_i, $i = 1,\ldots,n$. Such splits occur in practice with probability 0. However, they possess some very special and useful properties. We assume henceforth that x_i is fixed and suppress references to its identity in our various constructs. Thus, we write $u\ell$ for

$u\ell(x_i)$ and similarly for ur, TR, and TL. Note that $TL = (\ell_1, r_1, \ldots, \ell_i, 0)$ and $TR = (0, r_i, \ldots, \ell_n, r_n)$. Obviously, x_i is the last (i-th) minimum in ul and the first in ur. Recall that the local minima actually sampled in a given iteration is denoted $\bar{x}$.

Theorem 3.

(i) $$p_i(T) = p_1(TR) * P_i(TL).$$

(ii) $$p_j(T\colon \bar{x} \in \{x_1, \ldots, x_i\}) = p_j(TL), \text{ for every } j < i$$
$$p_j(T\colon \bar{x} \in \{x_i, \ldots, x_n\}) = p_j(TR), \text{ for every } j > i. \qquad \square$$

In words, (i) the probability of converging to x_i in u is equal to the product of the probabilities of doing so when the search is restricted to the left and right segments generated from u by a split at x_i. (ii) the conditional probability of converging to x_j given that we converge to a minimum to the left of x_i is equal to the unconditional probability of converging to that minimum when the search is restricted to $u\ell$, and similarly on the right.

Proof. Consider the sequence of points generated by the algorithm $Z = (z_1, z_2, \ldots)$ and the sequence of intervals $U = (u_1 \supseteq u_2, \ldots)$. It is convenient for the purposes of this proof to treat these sequences as infinite, i.e., to suspend the stopping rule $|u_k| < \delta$ and continue each iteration indefinitely. It is obvious that this convention does not change the probabilities we seek since when u_k reaches the length δ a unique local minimum has already been selected. Recall that z_k is uniform over u_k and that u_k depends on the initial segment of Z, $Z_{i-1} = (z_i, \ldots, z_{i-1})$. It follows that z_i depends on Z_{i-1} in a rather complex fashion. We can simplify the situation by partitioning the sequence Z into two subsequences $ZR = (zr_1, zr_2, \ldots)$ and $ZL = (z\ell_1, z\ell_2, \ldots)$ according to whether a given point is to the right or left of x_i. Also, we partition each interval u_k into two subintervals $ur_k = u_k \cap [x_i, b]$ and $u\ell_k = u_k \cap [a, x_i]$. Then $z_k \in ZR$ exactly if $z_k \in ur_k$ and vice versa for ZL. The probability of the former event is, of course, $|ur_k|/|u|$. A typical sequence $Z = (z_1, z_2, \ldots)$ may look like $Z = (zr_1, zr_2, z\ell_1, zr_3, \ldots)$, i.e., is composed of alternating subsequences of ZR and ZL. The evolution of the process Z could thus be described in the following equivalent way. First, choose one of the two intervals ur_k or $u\ell_k$ with probabilities $|ur_k|/|u|$ and $|u\ell_k|/|u|$, respectively. After this interval is chosen, pick z_k using the uniform distribution over that interval. z_k is classified as the next element of the ZR or ZL sequence in the obvious way. Finally, update the intervals ur_k and $u\ell_k$ as follows. If $z_k \in ur_k \cap L$ then $ur_{k+1} = ur_k \cap [z_k, b]$ and $u\ell_{k+1} = \emptyset$, and, if z_k, $ur_k \cap R$, then $ur_{k+1} = ur_k \cap [x_i, z_k]$ and $u\ell_{k+1} = u\ell_k$. Similarly, we handle the case of $z_k \in u\ell_k$. Note that the only dependence between the sequences ZL and ZR is through the interval update in the case $z_k \in ur_k \cap L$ or $z_k \in u\ell_k \cap R$. Call the first element which satisfies one of these conditions a <u>pivot</u>. Note that the pivot $z_k \in u\ell_k \cap R$ (a pivot the left) precisely when the process Z converges to a point strictly to the left of x_i and similarly to the right. The process converges to x_i precisely if no pivot occurs.

Define a process $W = (w_1, w_2, \ldots)$ which is closely related to Z but easier to analyze. Specifically, we start with intervals $v_1 = u_1$, $vr_1 = ur_1$, $v\ell_1 = u\ell_1$. At each step we choose vr_k or $v\ell_k$ with probabilities proportional to their length and pick w_k uniformly from the chosen interval. We include w_k in the WR sequence if it comes from vr_k and similarly for WL. The only new feature here is the update of vr_k and $v\ell_k$. If w_k is chosen from $vr_k \cap R$ then

$v\ell_{k+1} = v\ell_k$ and $vr_{k+1} = vr_k \cap [x_i, w_k]$ as is the case for the intervals U. However, if $w \in vr_k \cap L$ then we leave $v\ell_{k+1} = v\ell_k$ (instead of setting it to $\emptyset$ in the case of U) and set $vr_{k+1} = vr_k \cap [w_k, b]$. Similarly, we update only $v\ell_k$ if $w_k \in v\ell_k$. Naturally, the sequence W is not implementble algorithmically since x_i is typically unknown.

Clearly, the processes WL and WR are completely independent. Furthermore, the distribution of WL relative to u is identical to the distribution of Z relative to $u\ell$ and similarly for WR. Also, the processes W and Z are identical until the occurence of the first pivot. If that pivot is to the left, then Z continues to coincide with WL after this pivot. If the pivot is the right, then Z coincides with WR after the pivot.

We can now finalize the argument. For (i) note that the probability that both WR and WL converge to x_i is the right hand side of the assertion since the two processes are independent. Also, since Z and W are identical until the occurence of a pivot, the probability of a pivot occuring is identical in both processes. Finally, no pivot occurs in Z precisely if Z converges to x_i and no pivot occurs in W precisely if both WL and WR converge x_i. For (ii) observe that Z converges to a point to the left of x_i precisely if no pivot to the right occurs. $\square$

Remark. The reader may note that Theorem 3(ii) holds without any modification if the split is taken at a local maximum $y_i \in Y$.

An immediate but rather surprising corollary of Theorem 3(i) is the following crucial observation:

Corollary 3.1. $P_i(t_1, t_2 \ldots, t_{2n}) = P_i(0, t_2, \ldots, t_{2n-1}, 0)$.

In words, the sampling probabilities P are independent of the length of the first and last subintervals of T, ℓ_1 and r_n (t_1 and t_{2n}).

Proof. Consider a split at $x_1(x_2)$ and apply Theorem 3(ii). []

In the sequel we set, without loss of generality, $\ell_1 = r_n = 0$. Note that in this case $|u|$ is simply $x_n - x_1$. Applying the corollary to Formula (4), we get the following.

Theorem 4.

$$(5) \qquad p_i(T) = \frac{1}{x_n - x_1} [(\sum_{j=2}^{i} \ell_j p_{i-j+1}(TR(x_j)) + \sum_{j=1}^{n-1} r_j p_j(TL(x_j))] =$$

$$(6) \qquad = \frac{1}{x_n - x_1} [(\sum_{j=2}^{i} (x_j - y_j) p_{i-j+1}(0,r_j,\ldots,\ell_n,0) + \sum_{i}^{n-1} (y_{j+1} - x_j) p_i(0,r_1,\ldots,\ell_j,0))\ ] \quad []$$

(6) is a finite recursion which involves only subvectors of T. It can be used to calculate the entire vector P(T) in $O(n^3)$ operations (an $O(n^4)$ implementation is trivial). However, using Theorems 3 and 4 together, one can do much better. This will be discussed in the following section.

4. The Computation of P(T)

We are now in a position to obtain an explicit expression for P(T). The expression can be evaluated by an extremely simple algorithm which requires n computational steps. We start by examining the expression (6) for the special case i = 1. This yields

$$(7) \qquad P_1(0,r_1,\ell_2,\ldots,\ell_n,0) = \frac{1}{x_n - x_1} \sum_{j=1}^{n-1} (y_{j+1} - x_j) p_1(0,r_1,\ell_2,\ldots,\ell_j,0)$$

Define $q_j = p_1(0,r_1,\ell_2,\ldots,\ell_j,0)$ for $j = 2,\ldots,n$ $q_1 = 1$. Note that $P_1(T) = q_n$. Thus, (7) becomes

$$(8) \qquad q_n = \frac{1}{x_n - x_1} \sum_{j=1}^{n-1} (y_{j+1} - x_j) q_j$$

Applying the formula recursively we get for k = 1,...,n-1

$$(9) \qquad q_{k+1} = \frac{y_{k+1} - x_1}{x_{k+1} - x_1} q_k$$

This implies

$$(10) \qquad P_1(T) \equiv q_n = (\frac{y_n - x_1}{x_n - x_1})(\frac{y_{n-1} - x_1}{x_{n-1} - x_1}),\dots,(\frac{y_2 - x_1}{x_2 - x_1})$$

Similarly,

$$(11) \qquad P_n(T) = (\frac{x_n - y_2}{x_n - x_1})(\frac{x_n - y_3}{x_n - x_2}),\dots,(\frac{x_n - y_n}{x_n - x_{n-1}})$$

Finally, using Theorem 3(i) we get the required expression:

$$(12) \quad P_j(T) = [(\frac{y_2 - x_1}{x_2 - x_1})(\frac{y_3 - x_1}{x_3 - x_1}),\dots,(\frac{y_j - x_1}{x_j - x_1})] \; [(\frac{x_n - y_n}{x_n - x_1})(\frac{x_n - y_{n-1}}{x_n - x_{n-2}}), \dots,(\frac{x_n - y_j}{x_n - x_{j-1}})]$$

To compute P(T) effectively, first compute the expressions q_k, k = 1,...,n and their counterparts $\bar{q}_k$,

$$\bar{q}_k = (\frac{x_n - y_n}{x_n - x_{n-1}}),\dots,(\frac{x_n - y_{k+1}}{x_n - x_k}), \; k = n-1,\dots 2, \; \bar{q}_n = 1.$$

Each one of these expressions can be computed from its predecessor by a single multiplication in O(1) time. Thus, the entire series can be obtained in O(n) time. To get a specific value $p_i(T)$ we simply multiply the appropriate terms

$$p_i(T) = q_i \, \bar{q}_i, \; i = 1,\dots,n$$

Thus, the entire vector P(T) can be computed in O(n) time.

We can now compute P(T) for the examples cited in the introduction.

Example 1. ($y_1 = 0$, $x_1 = 1$, $y_2 = 5$, $x_2 = 6$, $y_3 = 10$). The two minima here may look symmetric, as T = (1,4,1,4). However, by corollary 3.1, we can replace t_1 and t_4 by 0 so that we can take T = (0,4,1,0). This is obviously a situation biased in favor of x_1. In fact, using (7) and (8) we get

$$p_1 = \frac{y_2 - x_1}{x_2 - x_1} = \frac{t_2}{t_1 + t_2} = 4/5$$

$$p_2 = \frac{x_2 - y_2}{x_2 - x_1} = \frac{t_3}{t_1 + t_2} = 1/5$$

Example 2. ($y_1 = 0$, $x_1 = 1$, $y_2 = 3$, $x_2 = 5$, $y_3 = 9$). This is an opposite example to Example 1. On first glance it appears that x_2 is more likely than x_1 as T = (1,2,2,4). However, using Corollary 3.1 we take T = (0,1,1,0) which yields $p_1 = p_2 = 1/2$.

Example 3. ($y_1 = 0$, $x_1 = 1$, $y_2 = 2$, $x_2 = 3$, $y_3 = 4$, $x_3 = 5$, $y_4 = 6$). Here T = (1,1,1,1,1,1) which is reduced to (0,1,1,1,1,0). Although the graph is symmetric, the probabilities are not equal. In fact, it is easy to obtain $q_1 = 1$, $q_2 = 1/2$, $q_3 = 3/8$. The $\bar{q}$'s assume the same values in the reverse order. Thus we get

$$p_1 = p_3 = q_3 = 3/8$$

$$p_2 = q_2 \bar{q}_2 = q_2^2 = 1/4.$$

The general symmetric case is handled in Section 6.

5. Properties

In this section we investigate some properties of the vector P which are directly relevant to the performance of random binary search in practice. Clearly, what we care about when the algorithm is actually applied is the probability $q = q(f,u)$ of picking the global optimum in one iteration as this probability is inversely related to the expected number of iterations required before that optimum is generated. We take the typical approach used in worst case analysis, i.e., we assume that the algorithm is applied to the "worst" instance in a certain class C of problem instances and seek to assess a bound $q(C)$ on $q(f,u)$ which is valid for all intances in the class C. Note that P depends on the vector T only, but that for a given T one can easily construct a function whose global minimum is any desired local minimum x_i, $i = 1,\ldots,n$. Thus, for each vector T the worst choice of f is such that $q = \min_{i=1,\ldots,n} P_i(T)$. An obvious class to examine is the class of functions with n or less local minima over the appropriate interval. Clearly in this case $q(C) < 1/n$ but as one would expect, it is not possible to derive a lower bound on q in terms of n. This is demonstrated by the example of figure 6 for the case $n = 3$.

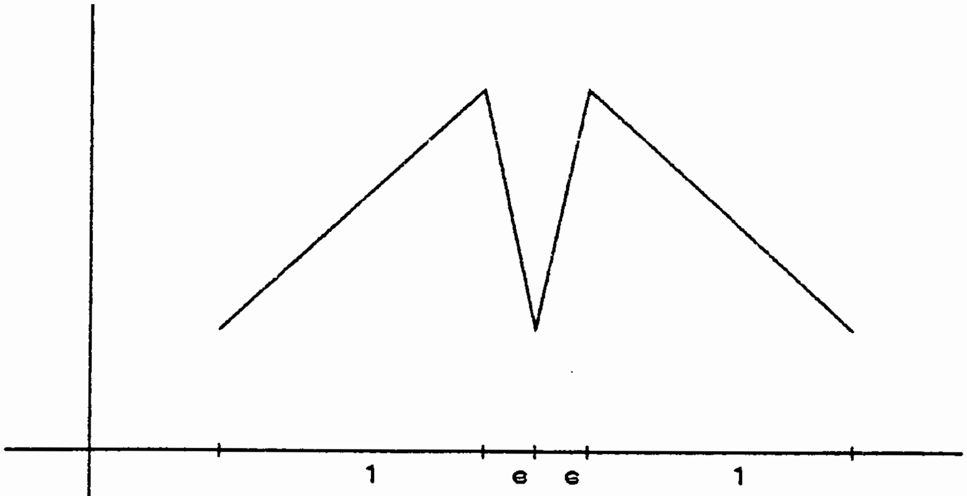

Figure 6

Here $T = (0,1,\varepsilon,\varepsilon,1,0)$, $p_2 = (\frac{\varepsilon}{1+\varepsilon})^2$ so that q can be made arbitrarily small by an obvious choice of ε. A more useful characterization of C is in terms of the minimal (normalized) distance between adjacent minima and maxima $\delta_0 = \frac{1}{|u|}(\min_{i=1,\ldots,n} \min\{r_i, \ell_i\})$, or the size of the smallest "valley" in u $\delta_1 = \frac{1}{|u|}(\min_{i=1,\ldots,n} \{r_i + \ell_i\})$. Ideally, one would have liked the probability of sampling a given minimum to equal the size of the appropriate "valley" as this would have meant that each subinterval of u is being sampled proportionally to its length and independently of its location inside u. This is, however, not the case as for a given size of δ_0 or δ_1, q can be arbitrarily small. (Examine figure 6 again and note that $\delta_0 = \frac{\varepsilon}{2(1+\varepsilon)}$ and $\delta_1 = \frac{\varepsilon}{(1+\varepsilon)}$). However, a <u>quadratic</u> lower bound on q in terms of δ_1 can be obtained:

<u>Theorem 5</u>. Let C be the class of problems with $\delta_1 > \Delta$. Then $q(C) = 4\Delta^2$.

<u>Proof</u>. (i) Examine the function depicted in figure 6 and choose ε such that $\frac{\varepsilon}{2(1+\varepsilon)} = \Delta$. This function clearly belongs to C. Also, $p_2(T)$ in that case is equal to $(\frac{\varepsilon}{1+e})^2 = 4\Delta^2$. Thus $q < 4\Delta^2$.

(ii) For any vector T and any minimum $x_i, P_i(T)$ is at least equal to the probability that the first z to the right of x_i is in r_i and the first z to its left is in ℓ_i. Thus,

$$P_i(T) > \frac{\ell_i \, r_i}{(x_i - a)(b - x_i)} > 4\Delta^2 \qquad \square$$

Theorem 5 means that some regions of u are undersampled relative to their size and consequently some are oversampled. In the next section we study this phenomenon more closely. One implication of the theorem is that if δ_0 is known (or could be estimated) and if one is concerned with <u>worst case</u>

analysis, then it is better to break u into $|u|/\delta_0 \equiv m$ intervals, each containing at most one minimum, and to apply regular (non-randomizing) binary search to each. This would require work proportional to m log m as opposed to $O(m^2 \log m)$ which is required to achieve a certain degree of confidence if the randomizing algorithm is used. Note, however, that the latter does not require any predetermination of δ_0 (or n) for its actual implementation. Also, we get a more favorable assessment of the performance of random binary search if we are ready to make some modest assumptions about the distribution of problem instances inside C. For instance, assume that a class C is given such that each problem instance in C has exactly n minima and such that for each vector T induced by an instance in C, all n possible local optima x_i, $i = 1,\ldots,n$ are equally likely to be the global optimum. Under these conditions, the probability q that the global minimum is sampled in a given iteration of the search is easily available.

Thereom 6. $q = 1/n$.

Proof. Pick any function of C and consider its T vector. Let $P(T) = (p_1(T),\ldots,p_n(T))$ be the sampling distribution of this vector. By our assumption, each minimum x_i, $i = 1,\ldots,n$ has equal probability of being the global minimum. Then

$$q = \sum_{i=1}^{n} p_i(T) \Pr[x = x_i] = 1/n. \qquad \square$$

We conclude this section with a certain desirable and intuitive monotonicity result. Consider the vectors T' and T related to each other by the relation $\ell_i = \ell_i'$, $i = 1,\ldots,n$, $r_i = r_i'$, $i = 1,\ldots,n$, $i \neq j$, $r_j > r_j'$, i.e., T is identical to T' except that the "valley" surrounding x_j in T is larger to the right. Surely, one expects $P_j(T) \geq P_j(T')$. This seems hard to show directly from (4) or (5), but falls easily out of (12).

Theorem 6. $P_i(T) \geq P_i(T')$ for $i = 1,\ldots,j$.

Proof. Direct for formula (12). □

6. The Symmetric Case

The probability $p_i(T)$ of sampling x_i depends on the relative sizes of the different "valleys" surrounding the local minima and on the "position" of x_i inside u. In this section we study the second effect by concentrating on completely symmetric vectors T, namely $t_i = 1$, $i = 1,\ldots,2n$. In spite of the symmetry in sizes, we will shortly discover that the various local minima have very different probabilities of being sampled.

Let T_n be the symmetric vector of 2n ones. Let $g_n(i) = p_i(T_n)$. We wish to study this function. Let $h(n) = p_1(T_n) = g_n(1)$. From Theorem 3 we have

$$g_n(i) = h(i)\ h(n - i + 1)$$

Thus, we first study the function h( ). From (9) we get the following simple recursion relation

$$h(n + 1) = \frac{2n - 1}{2n}\, h(n) \tag{13}$$

Clearly, $h(1) = 1$. Thus

$$h(n) = \frac{1}{2} \times \frac{3}{4} \times \frac{5}{6} \times \ldots \times \frac{2n - 3}{2n - 2}$$

We now wish to find a closed form approximation for h(n). Consider the function $f(n) = 1/\sqrt{n}$. Using the first term in the Taylor expansion of this function we get

$$\frac{1}{\sqrt{n+1}} = \frac{1}{\sqrt{n}} - \frac{1}{2n\sqrt{n}} + o(\frac{1}{n^2\sqrt{n}}) = \frac{1}{\sqrt{n}}(\frac{2n-1}{2n} + o(\frac{1}{n^2}))$$

that is

$$f(n+1) = (\frac{2n-1}{2n} + o(\frac{1}{n^2}))f(n).$$

with $f(1) = 1$.

Let $t(n) = h(n)/\sqrt{n}$. Then $t(n+1) = t_n(1 + o(\frac{1}{n^2}))$, $t(1) = 1$. Thus, for large values of n, $h(n)$ behaves approximately like $c/\sqrt{n}$ for some constant c (more precisely, one can show $c_1/\sqrt{n} \le h(n) \le c_2/\sqrt{n}$ and $\lim_{n\to\infty} h(n) = c/\sqrt{n}$ for some constants c_1, c_2 and c). Thus,

$$(14) \qquad \frac{c_1^2}{\sqrt{n-i+1}\,\sqrt{i}} \le g_n(i) \le \frac{c_2^2}{\sqrt{n-i+1}\,\sqrt{i}}$$

The actual values $g_n(i)$ together with their approximation $\frac{c}{\sqrt{n-i+1}\,\sqrt{i}}$ normalized to sum to 1 for $n = 30$ are shown in Figure 7 below.

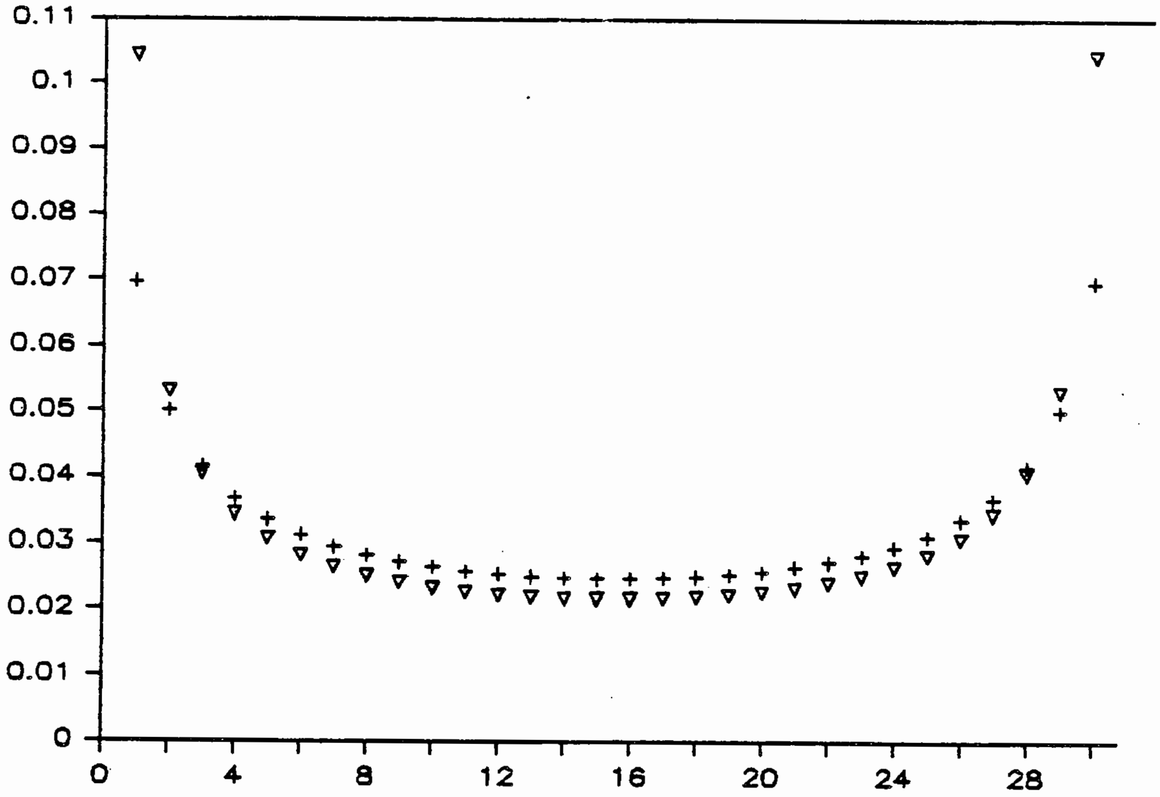

Figure 7. $g_n(i)$ for $n = 30$ (∇) and its approximation $\frac{c}{\sqrt{z}\,\sqrt{n-i}}$ (+)

For an asymptotically large n, we can consider the limiting behavior of the probabilities $p_i(T_n)$. Consider the following process defined on the interval $u = u_1 = [0,1]$. At each step, pick a point z_k uniformly distributed over $u_k = [a_k,b_k]$ and let u_{k+1} be $[a_k,z_k]$ or $[z_k,b_k]$ with probability 1/2, respectively. It can be shown that the process $z = (z_1,z_2,...)$ converges to a point $z \in [0,1]$ with probability one and that the density of the limit point z is given by

$$(15) \qquad f(z) = \frac{1}{\sqrt{z}\,\sqrt{1-z}}, \; 0 \leqslant z \leqslant 1$$

Clearly (15) is in agreement with (14). The implication of (14) or (15) is clear. Random binary search tends to oversample at the sides of the interval u and to undersample at the center. This may be helpful if the functions involved are likely to have their global optimum at the edges. If that is not the case, then this bias is bothersome. To offset it (but only partially) one can replace the uniform distribution over u_k by another distribution, heavier at the center and lighter at the tails. Alternatively, if the optima tend to be at the center, one may be better off breaking u into several segments and searching each separately using random binary search.

7. Adaptive Search

The simplest way to search the interval u using randomizing binary search is to apply each iteration independently to the interval u. This means that the information generated during the execution of the previous iterations is compeletly ignored. However, one can expect to do better if this information is somehow used to guide the search. In particular, say that the minimum located in the first iteration is x_1. One would certainly like to bias the search in the subsequent iterations such that the likelihood of x_1 being selected again is decreased. Naturally, the question arises whether this can be achieved. A simple way to go is the following. Split u at x_i and consider the two subintervals $ur(x_i)$ and $u\ell(x_i)$. (This can be done once x_i is identified in the first iteration.) Put these two intervals in a candidate list and attach a probability to each. At each iteration, pick an interval from the list according to its probability and apply random binary search to this interval. Then, split the subinterval at the new minimum found, modify the probabilities, and repeat the process. Leaving aside the question of how one assigns these probabilities, we examine below the question of whether this approach can be advantageous. Theorem 3 supplies an excellent vehicle for analyzing this issue.

Assume we split at x_i. let $p + q = 1$, and $p, q > 0$. Assume we chose ur with probability q and $u\ell$ with probability q and apply the search to the interval chosen. Let $Pr(i: p,q)$ be the probability of sampling x_i with this strategy. The question is whether we can pick p,q such that $Pr[i: p,q] < p_i(T)$. This is answered in the negative by

Theorem 8. $Pr[i: p,q] > p_i(T)$.

Proof.

$$Pr[i: p,q] = q \cdot p_1(TR) + p \cdot p_i(TL) > \min\{p_1(TR), p_i(TL)\} > p_1(TR) \cdot p_i(TL) = p_i(T) \quad \square$$

The question of whether one can devise other strategies for adaptive search which will decrease the possibilities of sampling x_i again is still open. We note that this issue is central to [3], [9], and [10], but in a very different context.

Acknowledgment

I wish to thank Isaac Meilijson for a very useful discussion concerning the solution (15) to the asymptotic process discussed at the end of section 6.

References

[1] Archetti, F. and B. Betro (1979), "A Probabilistic Algorithm for Global Optimziation," Calcolo 16, 335-343.

[2] Betro, B. (1982), "A Bayesian Algorithm for Global Optimization," paper presented at the International Institute for Statistics and Optimization, Gargagno, Italy.

[3] Boender, C. G. E., A. H. G. Rinnooy Kan, L. Stougie, and G. T. Timmer (1980), "Global Optimization: A Stochastic Approach," in F. Archetti amd M. Cugiani (Eds.), Numerical Techniques for Stochastic Systems, North-Holland, Amsterdam.

[4] Boender, C. G. E. (1984), The Generalized Multinomial Distribution: A Bayesian Analysis and Applications, Ph.D. disseration, Erasmus Universiteit Rotterdam (Centrum voor Wiskunde en Informatica, Amsterdam).

[5] Chandrasekaran, R. and A. Tamir (1982), "Polynomial Testing of the Query 'Is $a^b > c^d 2$?' with Application to Finding a Minimal Cost Reliability Spanning Tree," unpublished manuscript.

[6] Dixon, L. C. W. and G. P. Szego (Eds.) (1975), Towards Global Optimization, North-Holland, Amsterdam.

[7] Dixon, L. C. W. and G. P. Szego (Eds.) (1978), Towards Global Optimization 2, North-Holland, Amsterdam.

[8] Kuhm, H. W., Z. Wang and S. Xu (1984), "On the Cost of Computing Roots of Polynomials," Mathematical Programming 28, 156-164.

[9] Rinnooy Kan, A. H. G. and G. T. Timmer (1984), "Stochastic Methods for Global Optimization," to appear in the American Journal of Mathematical and Management Sciences.

[10] Timmer, G. T. (1984), "Global Optimization: A Stochastic Approach," Ph.D. thesis, Erasmus University Rotterdam, Rotterdam.

[11] Zemel, E., "Probabilistic Analysis of Geometric Location Problems," SIAM J. Alg. Dis. Meth. to appear.

[12] Zemel, E. (1983), "A Linear Time Randomizing Algorithm for Local Roots and Optima of Ranked Functions," unpublished manuscript.