



AgEcon SEARCH

RESEARCH IN AGRICULTURAL & APPLIED ECONOMICS

The World's Largest Open Access Agricultural & Applied Economics Digital Library

This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.

Help ensure our sustainability.

Give to AgEcon Search

AgEcon Search

<http://ageconsearch.umn.edu>

aesearch@umn.edu

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.

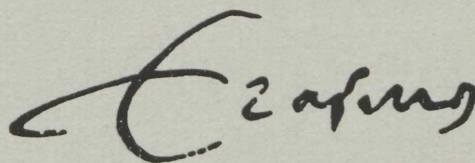
ECONOMETRIC INSTITUTE

STOCHASTIC GLOBAL OPTIMIZATION METHODS
PART II: MULTI LEVEL METHODS

A.H.G. RINNOOY KAN AND G.T. TIMMER

GIANNINI FOUNDATION OF
AGRICULTURAL ECONOMICS
LIBRARY
WITHDRAWN
OCT 2 1986

REPORT 8540/A



STOCHASTIC GLOBAL OPTIMIZATION METHODS

PART II: MULTI LEVEL METHODS

A.H.G. Rinnooy Kan* **

G.T. Timmer ** ***

ABSTRACT

In Part II of this paper, two stochastic methods for global optimization are described that, with probability 1, find all relevant local minima of the objective function with the smallest possible number of local searches. The computational performance of these methods is examined both analytically and empirically.

* Department of Industrial Engineering and Operations Research/Graduate School of Business Administration, University of California, Berkeley

** Econometric Institute, Erasmus University Rotterdam

*** ORTEC Consultants, Rotterdam

1. INTRODUCTION

In Part I of this paper [Rinnooy Kan & Timmer 1985], we described clustering methods for the global optimization problem

$$\min_{x \in S} f(x) \quad (1)$$

in which a strictly descent local search procedure P was applied to a subset of a sample drawn from a uniform distribution over S , so as to find all the local minima of f that are potentially global. In the k -th iteration of each method, the sample of size kN was first reduced by removing the $(1-\gamma)kN$ points with the highest function values. Three methods to deal with the reduced sample were presented (Density Clustering, Single Linkage and Mode Analysis), of which the last two turned out to have particularly attractive properties with respect to efficiency and accuracy. As far as the former is concerned, it could be shown that the number of local searches would remain finite (with probability 1), even if the sampling would continue indefinitely. With respect to the latter, we showed that (with probability 1) a local minimum would be detected in each level set component in which a point had been sampled. One such component may, however, contain several local minima; we are only assured of finding one of them. The purpose of this paper is to show how this final deficiency can be dealt with. Our notation will be the same as was used in Part I.

A crucial observation is that the methods described in Part I only make minimal use of the function values of the sample points. These function values are used to determine the reduced sample, but the clustering process applied to this reduced sample hardly depends on the function values. Instead, the clustering process concentrates on the location of the reduced sample points. As a result, the methods cannot distinguish between different regions of attraction which are located in the same component of $L(y_k^{(\gamma k N)})$. The function value of a sample point x can be of great importance if one wishes to predict to which region of attraction x belongs, because the local search procedure which defines these regions is assumed to be strictly descent. Hence, x cannot belong to the region of attraction of a local minimum x^* , if there is no descent path from x to x^* , i.e. a path along which the function values are monotonically decreasing.

Furthermore, x does certainly belong to the region of attraction of x^* , R_{x^*} , if there does not exist a descent path from x to any other minimum than x^* .

In Section 2, we describe two methods that exploit these observations, the Multi Level Single Linkage and Multi Level Mode Analysis method. These methods will be shown to invoke exactly one local search in each region of attraction (with probability 1). Thus, they guarantee the accuracy of the Multistart method (Section 2 of Part I) at the smallest possible cost in terms of local searches.

In Section 3, we turn to the computer implementation of the Multi Level methods. We shall show that they can be implemented in such a fashion that the expected computational effort up to iteration k is only linear in k , in spite of the fact that the entire sample is reconsidered in each iteration.

In Section 4, we present the results of computational experiments that confirm the speed and reliability of the Multi Level methods, as compared to other approaches to global optimization. Some concluding remarks are contained in the same section.

2. MULTI LEVEL METHODS

In the previous section, we observed that the proper assignment of a sample point x to a region of attraction R_{x^*} is related to the existence or nonexistence of a descent path from x to x^* . Obviously, it is impossible to consider all descent paths starting from x . Instead, we will (implicitly) consider all r_k -descent sequences, where a r_k -descent sequence is a sequence of sample points, such that each two successive points are within distance r_k of each other and such that the function values of the points in the sequence are monotonically decreasing. It will turn out that if the sample size increases and if r_k tends to 0, then every descent path can be conveniently approximated by such a sequence of sample points.

For a better understanding of the remainder of this section it is advantageous to consider the following algorithm first. Let w be the number of local minima known when the procedure is started.

Algorithm A.

Step 1. Initiate w different clusters, each consisting of one of the local minima present.

Step 2. Order the sample points, such that $f(x_i) < f(x_{i+1})$, $1 \leq i \leq kN-1$.
Set $i := 1$.

Step 3. Assign the sample point x_i to every cluster which contains a point within distance r_k .

If x_i is not assigned to any cluster yet, then start a local search at x_i to yield a local minimum x^* . If $x^* \notin X^*$, then add x^* to X^* , set $w := w+1$ and initiate the w -th cluster by x^* . Assign x_i to the cluster that is initiated by x^* .

Step 4. If $i = kN$, then stop. Else, set $i := i+1$ and go to Step 3.

Note that a sample point x can only be linked to a point with smaller function value that is within distance r_k (provided that a local search has not been applied unnecessarily, and the starting point is added to the resulting minimum for that reason only). Moreover (under the same provision), if x is assigned to a cluster which is initiated by a local minimum x^* , then there exists a r_k -descent sequence connecting x and x^* . The sample point x can be assigned to several clusters, if there exist r_k -descent sequences from x to each of the corresponding local minima.

Unfortunately, even if r_k tends to 0, then the fact that there exists an r_k -descent sequence from x to a local minimum x^* , does not necessarily imply that $x \in R_{x^*}$. If P is applied to x , then it is still possible that it will follow another descent path, and find another (possibly undetected) local minimum. However, as we will see later, this cannot happen if x is located in the interior of the basin of some local minimum.

Intuitively speaking, any two local minima will always be separated by a region with higher function values, so that Algorithm A will locate every local minimum in the neighbourhood of which a point has been sampled, if r_k is small enough. Since the function values are used in an explicit way in the clustering process, it is no longer necessary to reduce the sample. The method can be applied to the complete sample.

We could have named A Multi Level Single Linkage, since it is easy to check that it locates all local minima that can be obtained by applying

Single Linkage to the sample points with a function value smaller than $y_k^{(i)}$ for every $i = 1, 2, \dots, kN$. However, we reserve this name for an even simpler algorithm which locates the same minima. Note that it is not essential in Algorithm A to actually assign the sample points to clusters. For every sample point x , the decision whether P should be applied to x does not depend on the cluster structures; the decision only depends on the fact whether or not there exists a sample point z with $f(z) < f(x)$ within distance r_k of x . We now turn to an algorithm in which the superfluous clustering is omitted altogether! As in the case of Single Linkage it is necessary for the analysis of this algorithm that we avoid to start P in an element of Q_T or X_U^* (see Section 3 of Part I).

Multi Level Single Linkage

Step 1. For every $i = 1, 2, \dots, kN$ apply P to the sample point x_i except if $x_i \in Q_T \cup X_U^*$ or if there is a sample point x_j with $f(x_j) < f(x_i)$ and $\|x_j - x_i\| \leq r_k$.

Add new stationary points encountered during the local search to X^* .

Let us now analyze in how far Multi Level Single Linkage meets our initial goal, i.e. whether a local search is started if and only if a point is sampled in a region of attraction of a local minimum which has not been detected previously.

Let us first consider the probability that P is started incorrectly, in the sense that it results in a local minimum which was already known. Obviously, we can only start P in a sample point x_i if $x_i \in Y_U \setminus X_U^*$ or if $x_i \in M_{T,U}$ and there is no sample point x_j with $f(x_j) < f(x_i)$ within distance r_k of x_i . However, precisely this probability has been analyzed in Section 3.2 of Part I for the case that, for some $\sigma > 0$, r_k satisfies

$$r_k = \pi^{-\frac{1}{2}} \left(\Gamma \left(1 + \frac{n}{2} \right) m(S) \frac{\sigma \log kN}{kN} \right)^{1/n} \quad (2)$$

Hence, we can use the analysis of Section 3.2. of Part I to prove the next theorem.

THEOREM 1. If the critical distance r_k of Multi Level Single Linkage is determined by (2) with $\sigma > 0$, and if x is an arbitrary sample point, then the probability that P is applied to x by Multi Level Single Linkage tends to 0 with increasing k . If $\sigma > 2$, then the probability that a local search is applied by Multi Level Single Linkage in iteration k tends to 0 with increasing k . If $\sigma > 4$, then, even if the sampling continues forever, the total number of local searches ever started by Multi Level Single Linkage is finite with probability 1.

□

Now consider the second possible failure, i.e. the possibility that no local search is started in R_{x^*} , although a point has been sampled in R_{x^*} , and x^* has not been detected yet. As we remarked already, we cannot entirely exclude this possibility, since the existence of an r_k -descent sequence connecting a sample point x to a local minimum x^* does not necessarily imply that $x \in R_{x^*}$. However, we can prove that every local minimum will be found in the long run.

THEOREM 2. If r_k tends to 0 with increasing k , then, any local minimum x^* will be found by Multi Level Single Linkage within a finite number of iterations with probability 1.

PROOF. Consider the basin B_{x^*} of x^* . Recall that the basin is a component of $\{x \in S \mid f(x) < y_1\}$ that contains x^* as its only stationary point. Since y_1 is chosen such that $L_{x^*}(y_1)$ contains other stationary points as well (it equals S if there are no other stationary points) and since x^* is a local minimum, it follows that $y_1 > f(x^*)$.

Analogously to the proof of Theorem 11 of Part I it is easy to show that there exists a \bar{y} with $f(x^*) < \bar{y} \leq y_1$, such that the set $\{x \in B_{x^*} \mid f(x) < \bar{y}\}$ has positive measure and has an empty intersection with Q_T or $Y_U \setminus \{x \in S \mid \|x - x^*\| < \nu\}$. For any $\varepsilon > 0$, let E_ε be the set $\{x \in B_{x^*} \mid f(x) < \bar{y}_\varepsilon\}$, where \bar{y}_ε is the infimum of f over all points in B_{x^*} that are within distance ε of a point outside $\{x \in B_{x^*} \mid f(x) < \bar{y}\}$. Since $\bar{y} > f(x^*)$, it follows from the continuity of f that there exists a $\delta > 0$ such that E_δ has positive measure.

Suppose that a sample point exists in E_δ , and let x be the sample

point with the smallest function value in E_δ . Obviously, x cannot belong to Q_τ or $Y_\cup \setminus \{x \in S \mid \|x - x^*\| < \upsilon\}$. Moreover, it follows from the definition of E_δ and x that there cannot be a sample point z within distance δ of x with $f(z) < f(x)$. Hence, if $r_k < \delta$, then a local search will be started in x to find x^* , unless x^* has been discovered previously already. Since the probability that a sample point exists in E_δ tends to 0 with increasing k , the result is now immediate (cf. the analysis leading to Theorem 12, Part I).

□

Note that if we ignore the influence of the sets Q_τ and X_\cup^* , then \bar{y} equals y_1 in the proof of Theorem 2. Hence, in this case, we proved that a local minimum x^* will be discovered if $B_{x^*}(r_k)$ contains a sample point. (Recall that $B_{x^*}(r)$ is the set $\{x \in B_{x^*} \mid f(x) < \tilde{y}_r\}$, where \tilde{y}_r is the infimum over all points in B_{x^*} that are within distance r of an element outside B_{x^*} .) Since $B_{x^*}(r_k)$ tends to B_{x^*} if k tends to infinity, we obtain that in the limit, a local minimum x^* will be found if a point has been sampled in its basin.

The basin is the largest subset of R_{x^*} for which we can prove that if P is applied to one of its elements, it then converges to x^* . The reason is that for points outside a basin there may exist descent paths leading to different local minima. Intuitively speaking, the basin is the intersection of all regions of attraction belonging to the different possible strictly descent local search procedures. Hence we cannot really hope for substantially better methods than Multi Level Single Linkage if we do not further restrict the local search procedure in such a way to be able to translate this extra restriction into more information on the resulting regions of attraction.

We conclude this section with a method which we will call Multi Level Mode Analysis. This method is a generalization of Mode Analysis in a similar way and for the same reasons that Multi Level Single Linkage is a generalization of Single Linkage. As in Mode Analysis S is partitioned into cells, and we define a cell A to be full if it contains more than $\frac{1}{2}m(A)kN/m(S)$ reduced sample points. We define the function value of a cell A to be equal to the smallest function value of any of the sample points in A .

Multi Level Mode Analysis

- Step 1. (Determine reduced sample) Determine the reduced sample by taking the γkN sample points with the smallest function values.
- Step 2. (Define cells) Partition S into v cells.
- Step 3. (Determine full cells). For each cell A determine the number of reduced sample points in the cell. If this number exceeds $\frac{1}{2}m(A)kN/m(S)$, then the cell is full.
- Step 4. For every full cell A , determine the point \bar{x} which has the smallest function value among the reduced sample points in A . Apply P to \bar{x} except if $\bar{x} \in Q_T \cup X_U^*$, or if A has a neighbour which is full and has a smaller function value than $f(\bar{x})$. Add new stationary points encountered during the local search to X^* .

Note that we still reduce the sample. Although this is no longer strictly necessary, it creates the extra possibility that two regions of attraction are recognized as such only because a cell on the boundary of both regions is empty.

The cells are defined in the same way as they are defined in Mode Analysis, so that for some $\sigma > 0$, there are $kN/(\sigma \log kN)$ cells in iteration k .

Obviously, the arguments used to analyze the probability that a local search is started by Mode Analysis can also be used to prove the following theorem.

THEOREM 3. If, in iteration k , the number of cells in Multi Level Mode Analysis is $kN/(\sigma \log kN)$ with $\sigma > 10$, then the probability that a local search is started by Multi Level Mode Analysis in iteration k tends to 0 with increasing k . If $\sigma > 20$, then, even if the sampling continues forever, the total number of local searches ever started by Multi Level Mode Analysis is finite with probability 1.

□

If x^* is a local minimum for which $f(x^*) < y_\gamma$, then the proof of Theorem 2 can be easily adapted to show that the probability that Multi Level Mode Analysis will locate x^* tends to 1 with increasing k . If we define E_δ as in the proof of Theorem 2, then, for k large enough, the cell A containing x^* will be completely contained in E_δ . The probability that A

contains less than $\frac{1}{2}kNm(A)/m(S)$ sample points is again $O(k^{-\sigma/10})$. Moreover, since $f(x^*) < y_\gamma$ the probability that all sample points in A belong to the reduced sample tends to 1 with increasing k . We conclude that the probability that A is full tends to 1 with increasing k . Obviously, since $A \subset E_\delta$ the following theorem now follows from the definition of E_δ .

THEOREM 4. If, in iteration k , the number of cells in Multi Level Mode Analysis is $kN/(\sigma \log kN)$ with $\sigma > 0$, then, any local minimum x^* with $f(x^*) < y_\gamma$, will be found by Multi Level Mode Analysis within a finite number of iterations with probability 1. □

We conclude that the Multi Level methods and Multistart will result in the same set of minima with a probability which tends to 1 with increasing sample size. Hence, we did not go into the problem of designing a stopping rule for our methods. We simply suggest to use the stopping rules which were designed for Multistart (see Section 2 of Part I).

The assumptions under which the methods have been analyzed, were stated in Section 3 of Part I. Several of these assumptions concern the local search used. For instance, it is assumed that each local search converges to a local minimum and that the path followed by the search is completely contained in S . Although these assumptions simplify the exposition considerably, the methods proposed do not really depend on these assumptions. Obviously, if we drop both assumptions, then Multi Level Single Linkage and Multi Level Mode Analysis will still find the global minimum in the long run. This is due to the fact that there exists a neighbourhood (E_δ) of the global minimum which does not contain any other stationary points and which is a component of a certain level set. Therefore, the Multi Level methods ultimately start a local search in this neighbourhood to find the global minimum.

In this section we focused especially on the reliability of the methods considered. We did not pay much attention to the way in which they should be implemented. In the next section we describe some very efficient implementations of the more successful ones.

3. COMPUTER IMPLEMENTATION

So far, in our description of iterative global optimization methods, we have concentrated on one particular iteration, say k , and described the clustering procedures as if they were applied only once to a sample of size kN . Moreover, these static descriptions were focused primarily on the resulting number of local searches and not on implementation problems.

In this section, we will deal with the problem of efficient implementation of these methods. Of course, it is not efficient to start the calculations which are necessary for the clustering procedure from scratch in each iteration. Since the sample of iteration $k-1$ is a subset of the sample of iteration k , and since it is known in what way the parameters that define the clustering procedure (such as the critical distance) vary with k , it turns out to be possible to develop efficient dynamic implementations of the methods. In these dynamic implementations, the information which is necessary to determine the starting points of the local search procedure in iteration k will be determined by updating the corresponding information from iteration $k-1$. We will see that it is possible to implement Multi Level Single Linkage and Multi Level Mode Analysis in such a way that the running time needed up to iteration k is only $O(k)$ in expectation. This is surprisingly efficient: in each iteration, the entire sample is reconsidered so that a naive implementation would be $O(k^2)$.

There are two dynamic subproblems which need to be solved in most clustering procedures. Since most clustering methods use the reduced sample points only, we have to deal with the problem of determining this reduced sample, which is strongly related to the dynamic selection problem: find the γkN -th smallest function value in a sample of size kN for $k = 1, 2, \dots$. This problem is dealt with in detail in [Postmus et al. 1983], where we showed how to solve it in linear expected time. Actually, the analysis there is carried out only with respect to the number of comparisons required in the computation. Use of appropriate data structures such as doubly linked lists and 2-3 trees allows the conclusion to be extended to the running time in general.

A second subproblem which arises in clustering procedures such as Single Linkage, is that of testing whether or not there exists a sample

point z within the critical distance of a given point x (possibly with the extra restriction $f(z) < (f(x))$). This problem is related to the problem of finding nearest neighbours among uniformly distributed points which has been solved efficiently in [Bentley et al. 1980]. Below we discuss a data structure based on this reference, which allows for the determination of the nearest neighbour of a given point among kN uniform points in $O(1)$ expected running time.

In [Bentley et al. 1980] a technique is described which preprocesses kN uniform points in $O(k)$ running time, after which it is possible to determine the nearest neighbour of any query point (among the kN points) in $O(1)$ expected running time. We will first briefly describe this (static) technique and then show how it can be generalized for our purpose.

For some positive c , the idea of the preprocessing step is to assign each point to a small cell of measure $c/(kN)$ so that the expected number of points in each cell is c . We assume that $\sqrt[kN/c]{kN/c}$ is an integer so that we can choose the cells to be hypercubes (this assumption is not crucial). It is easy to check that this preprocessing can be performed in $O(k)$ running time. If we now wish to determine the nearest neighbour of some point x_1 , we first locate the cell containing x_1 . If this cell does not contain any sample point, then we start searching the cells surrounding it in an expanding pattern until a cell containing a sample point is found. Once we have this sample point, say x_2 , we are guaranteed that there is no need to investigate any cell that does not intersect with the hypersphere centered at x_1 of radius $\|x_2 - x_1\|$. The sample point nearest to x_1 can be found by considering all sample points in the cells that do intersect with this hypersphere. In [Bentley et al. 1980] it is proved that this Spiral Search finds the nearest neighbour of any given point in $O(1)$ expected running time.

Spiral Search can also be used to find the point closest to x_1 among the sample points with a function value smaller than $f(x_1)$. The only difference is that we now have to continue our spiralling search until we find a cell containing a sample point with a function value smaller than $f(x_1)$. If x_1 is an element of $M_{T,U}$, then we can use Theorem 7 in Part I to show that such an adapted version of Spiral Search finds the closest sample point with a function value smaller than $f(x_1)$ in $O(1)$ expected running time. To see why, let the i -th cell be the first cell in which a sample point is found with a function value smaller than $f(x_1)$. It is then easy to

check that the number of cells considered is $O(i)$, and that these cells contain $O(ci)$ sample points in expectation. From the Theorem 7 in Part I we know that there exists an r_0 such that for some $0 < \beta < \frac{1}{2}$

$$\frac{m(\{x \in S \mid \|x - x_1\| < r_0 \text{ and } f(x) < f(x_1)\})}{m(\{x \in S \mid \|x - x_1\| < r_0\})} > \beta.$$

Hence, the probability that no sample point exists in $\{x \in S \mid \|x - x_1\| < r_0 \text{ and } f(x) < f(x_1)\}$ decreases exponentially fast with k , so that we may ignore this possibility (the resulting costs are $O(k)$).

Obviously, the probability that an arbitrary cell in $\{x \in S \mid \|x - x_1\| < r_0\}$ does not contain a sample point with a function value smaller than $f(x_1)$ is bounded by $(1 - \beta c / (kN))^{kN}$ which is smaller than $e^{-\beta c}$, and the probability that the i -th cell considered is the first cell containing a sample point with a function value smaller than $f(x_1)$ decreases exponentially with i . Thus, we conclude that, for any $x_1 \in M_{T,U}$, Spiral Search finds the point which is closest to x_1 among the sample points with a function value smaller than $f(x_1)$ in $O(1)$ expected running time.

In Multi Level Single Linkage, we are only interested in finding the point closest to a point x_1 among the sample points with a function value smaller than $f(x_1)$ if such a point exists within the critical distance r_k of x_1 . Obviously, we can then stop the search if we have considered all cells that have a nonempty intersection with $A = \{x \in S \mid \|x - x_1\| \leq r_k\}$. For instance, if r_k is determined by (2), then $m(A) = (c \log kN) / (kN)$ and it is easy to check that we will not have to consider more than $O(\log k)$ cells.

To transform this static procedure into a dynamic one, it is suggested in [Bentley et al. 1980] to redefine the cells only if k is a power of 2, i.e. if $k = 2^m$ for some positive integer m . Hence, if $k = 2^m$, we insert all kN sample points in a new cell structure of $(kN)/c$ cells. If k is not a power of 2, then we do not change the cells, but only assign the N newly drawn points to the existing cells. Obviously, if we preprocess the points in this way, then, in any iteration, we can determine the nearest neighbour of an arbitrary point in $O(1)$ expected running time (the expected number of points in a cell is not always c , but is still bounded by $2c$).

Let us now consider the cost of the dynamic preprocessing of the sample points. In most iterations we only have to assign the N new sample points to the existing cells which takes no more than $O(1)$ running time.

If, for some integer m , $k = 2^m$, then we have to reinsert all kN sample points in the new cell structure which takes $O(k)$ running time. However, these latter iterations are rare enough not to prevent the running time up to iteration k to be $O(k)$ as well. To see why, note that if $k = 2^m$ for some integer m , then this implies that the previous $\frac{1}{2}k$ iterations were easy in the sense that they only took $O(1)$ running time per iteration. Hence, if we divide the $O(k)$ costs in iteration k over the $\frac{1}{2}k$ previous iterations, then we can say that the amortized running time per iteration is $O(1)$ (for a discussion of this notion of amortized cost, see [Tarjan 1983]). More precisely, if there exist constants c_1 and c_2 , such that a certain procedure takes c_1 calculations in iteration k if k is not a power of 2, and c_2k if k is a power of 2, then the number of calculations needed up to iteration k are smaller than $kc_1 + \sum_{m>0} c_2k2^{-m} \leq (c_1 + 2c_2)k$.

Now let us use this approach to design an efficient implementation of Multi Level Single Linkage. Given a set of stationary points X^* , Multi Level Single Linkage determines which of the kN sample points in iteration k should be used as starting points of the local search procedure, and updates the set X^* if new stationary points are encountered during these local searches. In the implementation of Multi Level Single Linkage described in this section, we use Spiral Search as a subroutine. If, for some point x , Spiral Search is applied to x in iteration k , then it will find the point closest to x among the sample points with a function value smaller than $f(x)$, unless no such sample point exists within the critical distance r_k of x . In the latter case, we say that Spiral Search did not succeed. We know that, after suitable preprocessing of the sample points, an application of Spiral Search to a sample point x in $M_{T,U}$ will take $O(1)$ running time in expectation, and that the number of cells considered during the search is $O(\log k)$ if r_k is determined by (2). The preprocessing of the sample points consists of partitioning S into small cells and assigning every sample point to the cell in which it is located. As indicated, we do not have to redefine the cells in every iteration. It suffices to define new cells if k is a power of 2 only. If k is not a power of 2 we simply assign the newly drawn points to one of the existing cells.

The outcomes of the applications of Spiral Search will be used to define lists $T(\ell)$ for every future iteration ℓ up to the next iteration in which the cells are redefined. These lists have the property that in

iteration k (k not a power of 2), $T(k)$ contains all sample points from which it may be necessary to start a local search.

Since we only redefine the cells if k is a power of 2, our implementation of Multi Level Single Linkage is divided in two parts depending on whether k is a power of 2 (case A) or not (case B). For a definition of notions like Q_T and X_U^* which we will use in the sequel, we refer to Part I. We will always choose the critical distance r_k according to (2).

A. Multi Level Single Linkage ($k = 2^m$, for some integer m)

Step A1. For some $c > 0$, partition S into $(kN)/c$ cells (hypercubes).

Step A2. Assign each sample point to the cell in which it is located.

Set $i:=0$

Step A3. Set $i:=i+1$. If i is greater than kN , stop.

If the i -th sample point, say x_i , is in $Q_T \cup X_U^*$, then repeat Step A3.

If x_i is in Y_U , then go to Step A4.

Apply Spiral Search to x_i . If the search does not succeed, then go to Step A4, else let z be the outcome of the search.

If $\|x_i - z\| > r_{2k}$, then determine the integer ℓ for which $r_{\ell-1} \geq \|x_i - z\| > r_\ell$ and add x_i to a list corresponding to iteration ℓ , say $T(\ell)$. Repeat Step A3.

Step A4. Apply P to x_i . Add new stationary points encountered during this search to X^* and go to Step A3.

B. Multi Level Single Linkage ($k \neq 2^m$, for any integer m)

Step B1. Assign the N new sample points to the existing cell structure; add these points to the list $T(k)$ and let t be the number of elements of the resulting list $T(k)$. Set $i:=0$.

Step B2. Set $i:=i+1$. If i is greater than t , stop.

If the i -th element of $T(k)$, say x_i , is in $Q_T \cup X_U^*$, then repeat Step B2.

If x_i is in Y_U , then go to Step B3.

Apply Spiral Search to x_i . If the search does not succeed, then go to Step B3, else let z be the outcome of the search.

If $\|x_i - z\|$ is greater than the critical distance of the next iteration in which the cells are redefined, then determine the integer ℓ for which $r_{\ell-1} \geq \|x_i - z\| > r_\ell$ and add x_i to $T(\ell)$. Repeat Step B2.

Step B3. Apply P to x_i . Add new stationary points encountered during this search to X^* and go to Step B2.

Apart from the set X^* , and the information concerning the sample points and the cells, we maintain a list $T(\ell)$ for every future iteration ℓ up to the next iteration k^+ in which the cells are redefined. If, in iteration k , we apply Spiral Search to a point x_i and find a point z with $f(z) < f(x_i)$ and $\|x_i - z\| \leq r_k$, and if $\|x_i - z\|$ is greater than the critical distance of iteration k^+ , then we add x_i to the list $T(\ell)$, where ℓ is the first iteration in which $\|x_i - z\|$ will no longer be smaller than or equal to the critical distance r_ℓ . Hence, it is easy to see that the list $T(\ell)$ ($k \leq \ell < k^+$) contains all sample points in $M_{\tau, \nu}$ from which no local search has been started yet and which have no sample point within distance r_ℓ with a smaller function value. Note that a list $T(\ell)$ may also contain other points, since we do not always know the nearest neighbour with a smaller function value of every sample point (updating this kind of information would be time consuming and is not necessary). However, if a sample point x in $M_{\tau, \nu}$ is not on the list $T(k)$ then we can be sure that either a sample point exists with $\|x - z\| \leq r_k$ and $f(z) < f(x)$ or that P has been applied to x previously already (provided that k is not a power of 2). Hence, if k is not a power of 2, then it suffices to consider the elements of $T(k)$ only (cf. Step B2).

It is easy to check that our implementation of Multi Level Single Linkage only differs from the previous description in the way the sample points in $Y_\nu \setminus X_\nu^*$ are treated. We now start P from every sample point in $Y_\nu \setminus X_\nu^*$, i.e. from every sample point that is within distance ν of a stationary point which has not yet been located. To be able to do so, we assume that it is possible to test whether a newly drawn point x is in $Y_\nu \setminus X_\nu^*$. This assumption is reasonable, since we can choose ν arbitrarily small. For instance, we can determine the gradient in x and decide that x is in Y_ν if this gradient is small. We treat the points in $Y_\nu \setminus X_\nu^*$ in this specific way because we cannot be sure that an application of Spiral Search

to a point in $Y_U \setminus X_U^*$ is possible in $O(1)$ expected running time. Since there are only a finite number of stationary points, we will only start a finite number of local searches from points in $Y_U \setminus X_U^*$, so that Theorem 1 and Theorem 2 are also valid for the above implementation of Multi Level Single Linkage.

Let us now examine the running time required by our implementation. First, suppose that k is a power of 2. We know that Step A1 and A2 only take $O(k)$ running time. In Step A3 we apply Spiral Search to elements x_i of $M_{\tau, \nu}$ only, so that the expected running time of each application is bounded by a constant. If Spiral Search finds a sample point z and $\|x_i - z\| > r_{2k}$, then we must determine the integer ℓ satisfying $r_{\ell-1} \geq \|x_i - z\| > r_\ell$. We cannot determine ℓ analytically, but it is obvious that ℓ can be determined by comparing $\|x_i - z\|$ and values r_j , for $O(\log k)$ values of j between k and $2k$ (e.g. by Binary Search). These $O(\log k)$ calculations only occur if $\|x_i - z\| > r_{2k}$, i.e. if no sample point z_1 exists with $f(z_1) < f(x_i)$ and $\|x_i - z_1\| \leq r_{2k}$ (else z would have been equal to z_1). The probability of this latter event, say α_k , is smaller than the probability that there is no sample point z_1 with $\|x_i - z_1\| \leq r_k^{\frac{\nu}{2}}$ and $f(z_1) < f(x)$ since $r_k^{\frac{\nu}{2}} < r_{2k}$. Since the measure of the set of points within distance $r_k^{\frac{\nu}{2}}$ of x_i is $(\frac{1}{2}\sigma \log k N)/(kN)$, for any $\beta < \frac{1}{2}$, we can use Theorem 7 of Part I to bound α_k by $O(k^{-\frac{1}{2}\beta\sigma})$. It follows that the expected running time needed for any sample point in Step A3 is $O(1)$, so that the expected running time for Step A3 is $O(k)$. Since the running time of a local search does not depend on k , we conclude that the expected running time for Step A1-A4, i.e. the expected running time for Multi Level Single Linkage in iteration k , where k is a power of 2, is $O(k)$.

Let us now examine the expected running time required in iteration k if k is not a power of 2. It is easy to adapt the analysis given for Step A1-A4 to show that the expected running time necessary for the treatment of the N new sample points can be bounded by a constant. Hence, it remains to consider the sample points that were already on the list $T(k)$ at the beginning of iteration k . We will prove that the expected running time required for Step B1-B3 in iteration k is $O(1)$ if $\sigma > 4$, by showing that, for any $\beta < \frac{1}{2}$, the expected number of sample points on the list $T(k)$ at the beginning of iteration k is $O(k^{1-\frac{1}{2}\beta\sigma})$ and that the expected running time to handle such a point is $O(\log k)$. To start with the latter, let x be an arbitrary point on the list $T(k)$ at the beginning of iteration k . We know

that in the iteration in which x was added to the list $T(k)$ there did not exist a sample point z with $f(z) < f(x)$ and $\|x-z\| \leq r_k$. Hence, we are no longer sure that the expected running time of an application of Spiral Search to x is $O(1)$ since we should condition on this knowledge. However, we can be sure that Spiral Search considers only $O(\log k)$ cells. The expected number of sample points in these cells is $O(\log k)$. Hence, for every sample point x which is on the list $T(k)$ at the beginning of iteration k we can bound the expected running time needed for an application of Spiral Search to x by $O(\log k)$. It is easy to check that the same bound is valid for the expected running time needed in Step B1-B3 for such a point x .

We shall now bound the expected number of sample points on the list $T(k)$ at the beginning of iteration k . Let x be any sample point in $M_{\tau, \nu}$ which has been sampled before iteration k (obviously these are the only possible candidates to be on the list $T(k)$ at the beginning of iteration k). Let k' be the most recent iteration in which Spiral Search was applied to x . Then $k' > \frac{1}{2}k$. If, in iteration k' , a sample point z did exist with $\|x-z\| \leq r_{\frac{1}{2}k} \sqrt{\frac{1}{2}}$ and $f(z) < f(x)$, then x cannot be on the list $T(k)$ (we already noticed that $r_k > r_{\frac{1}{2}k} \sqrt{\frac{1}{2}}$). The probability that there was no such sample point z in iteration k' is smaller than the probability that none of the $\frac{1}{2}kN$ sample points in iteration $\frac{1}{2}k$ was within distance $r_{\frac{1}{2}k} \sqrt{\frac{1}{2}}$ of x and had a function value smaller than $f(x)$. For any $\beta < \frac{1}{2}$, we can use Theorem 7 of Part I again to show that this latter probability is $O(k^{-\frac{1}{2}\beta\sigma})$. It follows that for any $\beta < \frac{1}{2}$ the expected number of elements on the list $T(k)$ at the beginning of iteration k is $O(k^{1-\frac{1}{2}\beta\sigma})$. Since the expected running time for each of these points is $O(\log k)$, we have proved that the expected running time of Step B1-B3, i.e. the expected running time for Multi Level Single Linkage in iteration k , where k is not a power of 2, is $O(1)$ if $\sigma > 4$. Combining the results for case A and B, we obtain the following theorem.

THEOREM 5. If the critical distance of Multi Level Single Linkage is determined by (2) with $\sigma > 4$, then it can be implemented in such a way that the expected running time up to iteration k is $O(k)$ and the amortized expected running time is $O(1)$ in every iteration.

□

The implementation of Multi Level Single Linkage described so far can be easily adapted to handle the situation in which one wishes to use the reduced sample only.

Let us now describe an efficient implementation of Multi Level Mode Analysis. In the first description of this method we defined the number of cells in iteration k to be equal to $kN/(\sigma \log kN)$, and we defined a cell to be full if it contained more than $\frac{1}{2}\sigma \log kN$ reduced sample points, i.e. more than half the expected number of sample points in the cell. Obviously, the redefinition of the cells in every iteration is prohibitive for a very efficient implementation. As before, we will only define new cells if k is a power of 2. In iteration k the number of cells in our implementation is equal to $k^-N/(\sigma \log k^-N)$, where k^- is the most recent iteration number which is a power of 2. The expected number of sample points in iteration k is then equal to $(k\sigma \log k^-N)/k^-$, and we will now define a cell to be full if it contains more than $(\frac{1}{2}k\sigma \log k^-N)/k^-$ reduced sample points. Since the number of cells in iteration k , $k^-N/(\sigma \log k^-N)$, is between $\frac{1}{2}kN/(\sigma \log kN)$ and $kN/(\sigma \log kN)$, it is easy to check that this adaptation of the method does not affect the theoretical results obtained in Section 2.

Between iteration k^- , which is a power of 2, and iteration $k^+ = 2k^-$, we do not change the cells but only update information concerning the cells. In each cell (say A_i) we iteratively update (if necessary) the reduced sample point with the smallest function value (say \bar{x}_i) and the total number of reduced sample points in the cell (say a_i). Apart from the information concerning the cells we will also maintain a list $T(\ell)$ for every future iteration ℓ up to iteration k^+ . Such a list $T(\ell)$ will contain every cell which, after being full in iteration $\ell-1$, will become empty in iteration ℓ , provided that no points are added to or removed from the cell until iteration ℓ . We iteratively update these lists to prevent that we have to check every cell in every iteration.

If, in iteration k , a cell is on the list $T(k)$, then we add all its neighbours to an initially empty list, say G . The cells which become full in iteration k (after being empty in iteration $k-1$) and the cells for which \bar{x}_i changes in iteration k are also added to the list G . As we shall see later the list G then contains all cells for which it may be necessary to apply P to one of its elements.

Finally, we note that, as in the previous sections, we will not go

into the details of how to store the cell structure itself. It is easy to see that we can number the cells such that, for each sample point we can find the cell in which it is located in $O(1)$ running time, and for each cell we can find all neighbouring cells in $O(1)$ running time as well.

A. Multi Level Mode Analysis ($k = 2^m$ for some integer m)

Step A1. Determine the reduced sample.

Step A2. Partition S into $kN/(\sigma \log kN)$ cells. Set $i := 0$

Step A3. Set $i := i+1$. If i is larger than the number of cells, then set $i := 0$ and go to Step A4.

Consider the i -th cell, say A_i . Determine the point \bar{x}_i which has the smallest function value among the reduced sample points in A_i , and determine the number of reduced sample points in A_i say a_i .

Step A4. Set $i := i+1$. If i is greater than the number of cells, stop.

If a_i does not exceed $\frac{1}{2}\sigma \log kN$, then repeat Step A4.

Let iteration ℓ be the first iteration in which a_i elements are not enough for a cell to be full, i.e. ℓ is the smallest integer greater than or equal to $2a_i k/(\sigma \log kN)$. If $\ell < 2k$, then add A_i to the list $T(\ell)$.

Apply P to \bar{x}_i , except if $\bar{x}_i \in Q_\tau \cup X_\cup^*$ or if A_i has a neighbour which contains more than $\frac{1}{2}\sigma \log kN$ sample points, at least one of which has a function value smaller than $f(\bar{x}_i)$. Add new stationary points encountered during a local search to X^* .

B. Multi Level Mode Analysis ($k \neq 2^m$ for any integer m)

Step B1. Determine the sets D_1 and D_2 of points entering, respectively leaving, the reduced sample. Let the number of elements of D_1 and D_2 be d_1 and d_2 respectively. Let k^- be the most recent iteration number which was a power of 2, and let k^+ equal $2k^-$. Set $j = 0$ and $p := 0$.

Step B2. Set $j := j+1$. If j is greater than d_1 , then go to step B3.

Determine the cell $A_{i(j)}$ in which the j -th elements of D_1 is located. Set $a_{i(j)} := a_{i(j)} + 1$. If the function value of the j -th element of D_1 is smaller than $f(\bar{x}_{i(j)})$, then replace $\bar{x}_{i(j)}$ by this element and if in addition $A_{i(j)}$ is full

(i.e. $a_{i(j)} > (\frac{1}{2}\sigma k \log k^{-N})/k^{-}$), then add $A_{i(j)}$ to an initially empty list G .

If not already on the list, add $A_{i(j)}$ to the list G if $A_{i(j)}$ was empty in the previous iteration and has now become full, i.e. if $a_{i(j)}^{-1} \leq (\frac{1}{2}\sigma(k-1)\log k^{-N})/K^{-}$ and if $a_{i(j)} > (\frac{1}{2}\sigma k \log k^{-N})/k^{-}$.

Let ℓ and ℓ' be the smallest integers greater than or equal to $2(a_{i(j)}^{-1})k^{-}/(\sigma \log k^{-N})$ respectively $2a_{i(j)}k^{-}/(\sigma \log k^{-N})$.

If $\ell' = \ell$, then repeat Step B2.

If $k \leq \ell < k^{+}$, then delete $A_{i(j)}$ from the list $T(\ell)$.

If $k < \ell' < k^{+}$, then add $A_{i(j)}$ to the list $T(\ell')$. Repeat Step B2.

Step B3. Set $p := p+1$. If p is larger than d_2 , then go to Step B4.

Determine the cell $A_{i(p)}$ in which the p -th element of D_2 is located. Set $a_{i(p)} := a_{i(p)}^{-1}$.

Let ℓ and ℓ' be the smallest integer greater than or equal to $2(a_{i(p)}+1)k^{-}/(\sigma \log k^{-N})$ and $2a_{i(p)}k^{-}/(\sigma \log k^{-N})$ respectively.

If $\ell \leq k$, then repeat Step B3.

If $\ell < k^{+}$, then delete $A_{i(p)}$ from the list $T(\ell)$.

If $k \leq \ell' < k^{+}$, then add $A_{i(p)}$ to the list $T(\ell')$.

If $\ell' < k$, then add $A_{i(p)}$ to the list $T(k)$.

Repeat Step B3.

Step B4. For every cell on the list $T(k)$, add all neighbours of the cell to the list G , provided that they are full and not already on this list.

Step B5. For every cell on the list G , apply P to the reduced sample point x with the smallest function value in the cell except if $x \in Q_{\tau} \cup X_{\cup}^{*}$ or if the cell has a neighbour that contains more than $(\frac{1}{2}k\sigma \log k^{-N})/k^{-}$ reduced sample points, at least one of which has a function value smaller than $f(x)$. Add new stationery points encountered during a local search to X^{*} .

It is not too difficult to verify that Theorems 4 and 5 apply to the above implementation. We omit the details and conclude this section by considering the expected running time needed by our implementation in iteration k . Using the dynamic selection method of [Postmus et al. 1983], we can perform Step A1 and Step B1 in $O(1)$ expected running time. If k is a power of 2, then it is easy to check that Step A2 and A3 can be performed in $O(k)$ running time. Since every cell does only have $O(1)$ neighbours,

every cell can be treated in $O(1)$ running time in Step A4. It follows that Step A1-A4 only take $O(k)$ running time in expectation.

Now suppose that k is not a power of 2. Each element of D_1 will only cause $O(1)$ running time in Step B2 and can only cause one cell to be added to the list G . Similarly, each element of D_2 can be treated in $O(1)$ running time in Step B3. Since d_1 and d_2 are both smaller than or equal to N , it follows that Step B2 and B3 can be performed in $O(1)$ running time, and that no more than N cells can be on the list G at the beginning of Step B4. Obviously, each cell on the list $T(k)$ can only cause $O(1)$ cells to be added to G and each cell on the list G can be treated in $O(1)$ running time in Step B5. Hence, to prove that Step B1-B5 can be performed in $O(1)$ expected running time, it suffices to show that the list $T(k)$ contains $O(1)$ cells at the beginning of Step B4.

A cell can only be on the list $T(k)$ at the beginning of Step B4, if it was full in iteration k and has now become empty. It is easy to see that the expected number of full cells does not change from iteration $k-1$ to iteration k (provided that k is not a power of 2 and the cells are redefined). Moreover, a cell can only change from empty in iteration $k-1$ to full in iteration k if a point is added to the cell in iteration k . Since d_1 is at most equal to N , it follows that the expected number of cells that change from empty to full in iteration k is at most N . Combining this last result with the fact that the expected number of full cells in iteration k is equal to the corresponding number in iteration $k-1$, it follows that the expected number of cells on the list $T(k)$ at the beginning of Step B4 is also at most equal to N . Hence, we proved the following theorem.

THEOREM 6. Multi Level Mode Analysis can be implemented such that the expected running time up to iteration k is $O(k)$ and the amortized expected running time is $O(1)$ in every iteration.

□

4. COMPUTATIONAL EXPERIMENTS

In this section we shall discuss the computational performance of the methods described in Part I and Part II on a number of test problems. For this purpose the algorithms were coded in Fortran IV and run on the DEC 2060 computer of the Computer Institute Woudestein.

To be able to compare our methods with other existing ones, the unconstrained methods have been tested on the standard set of test functions [Dixon & Szegö 1978b], which is commonly used in global optimization. (see Table 1) Since all test functions are twice continuously differentiable, we used the VALOAD variable metric subroutine from the Harwell Subroutine Library as the local search procedure in all experiments.

After some initial experiments we concentrated on Single Linkage, Multi Level Single Linkage and Multi Level Mode Analysis. To obtain an impression of the numerical performance of these procedures, we applied them to four independent samples of size 1000. For all three methods we reduced the sample to 100 points ($\gamma=0.1$). Furthermore, we chose both ν and τ to be equal to zero in all experiments, thus neglecting the sets Q_τ and X_ν^* . In Multi Level Single Linkage and in Single Linkage we set σ equal to 4. In Single Linkage we used the gradient criterion (Section 4 of Part I) in an attempt to approximate the regions of attraction instead of the components of a level set. In Multi Level Mode Analysis we encountered the problem that if the sample size is 1000 then the number of cells is still not very large. Especially in higher dimensional problems each cell is a neighbour of most other cells. We therefore chose σ to be only one and restricted the definition of neighbouring cells. In our implementation we said two cells to be neighbours if they have an $(n-1)$ -dimensional facet in common, so that a cell has at most $2n$ neighbours.

The average results of the four runs are listed in Table 2.

Table 1

TESTFUNCTIONS [Dixon & Szegö 1978b]

GP	Goldstein & Price
BR	Branin (RCOS)
H3	Hartman 3
H6	Hartman 6
S5	Shekel 5
S7	Shekel 7
S10	Shekel 10

Table 2

Samples of size 1000

		Single Linkage	Multi Level Single Linkage	Multi Level Mode Analysis
Function				
GP:	l.m.	3	3	3
	l.s.	3	3	5
	f.e.	163	91	117
BR:	l.m.	3	3	3
	l.s.	3	3	3
	f.e.	157	65	63
H3:	l.m.	2	2	2
	l.s.	2	4	3
	f.e.	161	112	106
H6:	l.m.	2	2	2
	l.s.	5	10	5
	f.e.	585	986	454
S5:	l.m.	5	5	4
	l.s.	5	5	5
	f.e.	324	211	214
S7:	l.m.	6(*)	6(*)	5(*)
	l.s.	6	6	5
	f.e.	429	281	224
S10:	l.m.	7(*)	8(*)	5(*)
	l.s.	7	8	5
	f.e.	439	346	238

l.m.: number of local minima found

l.s.: number of local searches performed

f.e.: number of function evaluations required (not including the 1000
function evaluations required to determine the function values of the
sample points)

(*) : global minimum was not found in one of the four runs

In one of the four runs the methods did not find the global minimum of both the S7 and the S10 test. The reasons for this are twofold. Firstly, the global minimum of these functions is relatively close to other local minima. Secondly, one of the four samples happened to be a very unfortunate one: the regions of attraction surrounding the region of attraction of the global minimum contained sample points whose function values were smaller than the smallest function value attained in a sample point in the region of attraction of the global minimum.

The number of local searches started unnecessarily is the largest for the test functions H3 and H7. This is due to the fact that these functions are badly scaled. The results obtained on these functions can be improved considerably using the Hessians at the local minima found.

On the basis of Table 2, we decided to continue the computational experiments with Multi Level Single Linkage. This method has been compared with a few leading contenders whose computational behaviour is described in [Dixon & Szegö 1978b]. In this reference methods are compared on the basis of two criteria: the number of function evaluations and the running time required to solve each of the seven test problems. To eliminate the influence of the different computer systems used, the running time required is measured in units of standard time, where one unit corresponds to the running time needed for 1000 evaluations of the S5 test function in the point (4,4,4,4).

Since both the number of function evaluations and the units of standard time required are sensitive to the peculiarities of the sample at hand, the results reported for Multi Level Single Linkage are the average outcome of four independent runs again. As before we chose $\tau = \nu = 0$ and $\sigma = 4$ in our implementation of Multi Level Single Linkage. However, we now applied Multi Level Single Linkage to 20% of the sample points ($\gamma = 0.2$) (the reason that we set γ equal to 0.1 before was that a major reduction of the sample is necessary for successful application of Single Linkage). Furthermore, it did not seem reasonable to apply Multi Level Single Linkage to samples of fixed size. After an initial sample of size 100, we increase the sample and applied Multi Level Single Linkage iteratively until the expected number of minima (in the Bayesian sense - see Theorem 2, Part I) exceeded the number of different minima found by less than 0.5.

In Tables 4 and Table 5 we summarize the computational results of the methods listed in Table 3 (except for Multi Level Single Linkage, the results are taken from [Dixon & Szegö 1978b]).

Table 3

Methods

-
- A Trajectory method [Branin & Hoo 1972].
- B Random direction method [Bremmerman 1970].
- C Controlled Random Search [Price 1978].
- D Method proposed in [Törn 1976, 1978] based on concentration of the sample and density clustering.
- E Method based on reduction, density clustering and a spline approximation of the distribution function ϕ of f [De Biase & Frontini 1978].
- F Multi Level Single Linkage.
-

Table 4

Number of function evaluations

Method	Function						
	GP	BR	H3	H6	S5	S7	S10
A	-	-	-	-	5500	5020	4860
B	300	160	420L	515	375L	405L	336L
C	2500	1800	2400	7600	3800	4900	4400
D	2499	1558	2584	3447	3649	3606	3874
E	378	597	732	807	620	788	1160
F	148	206	197	487	404	432(*)	564

L : the method did not find the global minimum

(*) : the global minimum was not found in one of the four runs

Table 5

Number of units standard time

Method	Function						
	GP	BR	H3	H6	S5	S7	S10
A	-	-	-	-	9	8.5	9.5
B	0.7	0.5	2L	3	1.5L	1.5L	2L
C	3	4	8	46	14	20	20
D	4	4	8	16	10	13	15
E	15	14	16	21	23	20	30
F	0.15	0.25	0.5	2	1	1 ^(*)	2

L : the method did not find the global minimum

(*) : the global minimum was not found in one of the four runs

As before Multi Level Single Linkage did not find the global minimum of the S7 test function in one of the four runs. This failure could have been prevented by choosing σ to be equal to 2. In that case, the computational results of the method obtained on the test functions GP, BR, H3, H7 and S5 turn out to be comparable to the numbers given in Table 4 and Table 5. However, the number of function evaluations required for the functions S7 and S10 increase by a factor of 2 and 3 respectively. This is due to the fact that all minima of both functions are found in an early stage if σ equals 2. However, the sample must then be increased considerably before our stopping criterion is satisfied.

Since the stopping rules involved in the methods listed in Table 3 are totally different, the comparison between the methods can never be entirely fair: there is always a trade-off between reliability and computational effort that is hard to measure consistently. However, we feel confident that Multi Level Single Linkage is one of the most reliable and efficient methods presently available.

The most important disadvantage of the methods dealt with in Part I and Part II is that their performance still depends heavily on the dimension n of the space over which the minimization is performed. (This problem did not arise in our analysis, since we always treated n as a constant.) The problem is most obvious in the Mode Analysis approach since in high dimensional spaces all cells will be neighbours of each other except if the sample is very large. But in (Multi level) Single Linkage the problem is equally existing.

This 'curse of dimensionality' [Dixon et al. 1976] is present in all efficient methods that have been proposed for global optimization. A reason to believe that it could be countered in principle is that the Bayesian analysis described in Section 2 of Part I does not depend on n . Apart from the fact that a local search procedure may be less efficient in higher dimensional space, the only parameters which influence the behaviour of Multistart are the number of local minima and the relative sizes of its regions of attraction. Moreover, the number of local searches required to find the global minimum only depends on the relative size of the region of attraction of the global minimum which does not necessarily depends on n . However, to avoid the problems occurring in high dimensional spaces an approach basically different from the one described here seems to be necessary.

Another way in which the methods might be improved is suggested in the observation that it seems natural to change the sampling distribution as a result of what has been learned in previous iterations. Most results can probably be generalized to hold for a wide class of sampling distributions. It seems difficult, however, to adapt the Bayesian analysis and the resulting stopping rules so as to hold for distributions different from the uniform one.

We conclude that the approach suggested to solve the global optimization problem has many attractive properties, but that further research will be necessary before practical global optimization problems, especially in high dimensional spaces, can be solved with true efficiency. Thus, it seems that global optimization, an area of research which has been neglected for such a long time, will continue to offer interesting challenges to researchers from widely differing backgrounds.

ACKNOWLEDGEMENTS

The research of the first author was partially supported by a NATO Senior Scientist Fellowship. The research of the second author was partially supported by the Netherlands Foundation for Mathematics SMC with financial aid from the Netherlands Organization for Advancement of Pure Research (ZWO).

REFERENCES

- Bentley, J.L., B.W. Weide and A.C. Yao (1980), Optimal expected-time algorithms for closest point problems. ACM Transactions on Mathematical Software 6, 563-580.
- Branin, F.H. and S.K. Hoo (1972), A method for finding multiple extrema of a function of n variables. In F.A. Lootsma (ed.), Numerical Methods of Nonlinear Optimization (Academic Press, London).
- Bremmerman, H. (1970), A method of unconstrained global optimization. Mathematical Biosciences 9, 1-15.
- De Biase, L. and F. Frontini (1978), A stochastic method for global optimization: its structure and numerical performance. In [Dixon & Szegö 1978a].
- Dixon, L.C.W., J. Gomulka and S.E. Hersom (1976), Reflections on the global optimization problem. In L.C.W. Dixon (ed.), Optimization in Action (Academic Press, London).
- Dixon, L.C.W. and G.P. Szegö (eds.) (1978a), Towards Global Optimization 2 (North-Holland, Amsterdam).
- Dixon, L.C.W. and G.P. Szegö (1978b), The global optimization problem. In [Dixon & Szegö 1978a].
- Postmus, J.T., A.H.G. Rinnooy Kan and G.T. Timmer (1983), An efficient dynamic selection method. Communications of the ACM 26, 878-881.
- Price, W.L. (1978), A controlled random search procedure for global optimization. In [Dixon & Szegö 1978a].
- Rinnooy Kan, A.H.G. and G.T. Timmer, 'Stochastic global optimization methods. Part I: clustering methods'.
- Tarjan, R.E. (1983), Data Structures and Network Algorithms. Siam CBNS/NSF Regional Conference Series in Applied Mathematics.
- Törn, A.A. (1976), Cluster analysis using seed points and density determined hyperspheres with an application to global optimization. In Proceeding of the third International Conference on Pattern Recognition, Coronado, California.
- Törn, A.A. (1978), A search clustering approach to global optimization. In [Dixon & Szegö 1978a].

LIST OF REPORTS 1985

- 8500 "Publications of the Econometric Institute Second Half 1984: List of Reprints 378-400, Abstracts of Reports".
- 8501/0 R.M. Karp, J.K. Lenstra, C.J.H. McDiarmid and A.H.G. Rinnooy Kan, "Probabilistic analysis of combinatorial algorithms: an annotated bibliography", 26 pages.
- 8502/E M.E. Homan, "Verschillen in consumptie tussen één en tweekostwinnerhuishoudens: een eerste analyse", 13 pages.
- 8503/0 A.W.J. Kolen, "The round-trip p-center and covering problem on a tree", 17 pages.
- 8504/0 H.C.P. Berbee, C.G.E. Boender, A.H.G. Rinnooy Kan, G.L. Scheffer, R.L. Smith and J. Telgen, "Hit-and-run algorithms for the identification of nonredundant linear inequalities", 32 pages.
- 8505/S H. Brozius and L. de Haan, "On limiting laws for the convex hull of a sample", 10 pages.
- 8506/0 J.B.G. Frenk and A.H.G. Rinnooy Kan, "On the rate of convergence, to optimality of the LPT rule - postscript", 5 pages.
- 8507/E P. Kooiman, H.K. van Dijk and A.R. Thurik, "Likelihood diagnostics and Bayesian analysis of a micro-economic disequilibrium model for retail services", 35 pages.
- 8508/0 C.G.E. Boender, A.H.G. Rinnooy Kan and J.R. de Wit, "A Bayesian procedure for the (s,Q) inventory problem", 26 pages.
- 8509/E B.M.S. van Praag, S. Dubnoff and N.L. van der Sar, "From judgments to norms: measuring the social meaning of income, age and education", 35 pages.
- 8510/S B. Bode, J. Koerts and A.R. Thurik, "On shopkeepers' pricing behaviour", 19 pages.
- 8511/M H. Bart, I. Gohberg and M.A. Kaashoek, "Exponentially dichotomous operators and inverse Fourier transforms", 70 pages.
- 8512/M J. Brinkhuis, "Normal integral bases and embedding of fields", 13 pages.
- 8513/E P.M.C. de Boer, "On the relationship between Revankar's and Luffletcher's production function", 5 pages.
- 8514/S L. de Haan, "Extremes in higher dimensions: the model and some statistics", 15 pages.
- 8515 Publications of the Econometric Institute First Half 1985: List of Reprints 401-414, Abstracts of Reports.

- 8516/A **A.R. Thurik and A. Kleijweg**, "Cyclical effects in retail labour productivity", 20 pages.
- 8517/C **B.M.S. van Praag and J. van Weeren**, "The impact of past experiences and anticipated future on individual income judgements", 24 pages.
- 8518/A **A.R. Thurik and J. Koerts**, "Behaviour of retail entrepreneurs". 16 pages.
- 8519/B **M. Hazewinkel, J.F. Kaashoek and B. Leynse**, "Pattern formation for a one dimensional evolution equation based on Thom's River Basin Model". 24 pages.
- 8520/A **H.K. van Dijk, T. Kloek and C.G.E. Boender**, "Posterior moments computed by mixed integration", 25 pages.
- 8521/B **J. Brinkhuis**, "Testing concavity and quasi-concavity is easy", 12 pages.
- 8522/A **R. Harkema**, "Minimum sample size requirements for maximum likelihood estimation of some demand models", 25 pages.
- 8523/B **A.C.F. Vorst**, "The general linear group of discrete Hodge algebras", 10 pages.
- 8524/A **A.M.H. Gerards and A.W.J. Kolen**, "Polyhedral combinatorics in combinatorial optimization", 25 pages.
- 8525/A **B.J. Lageweg, J.K. Lenstra, A.H.G. Rinnooy Kan and L. Stougie**, "Stochastic integer programming by dynamic programming", 19 pages.
- 8526/A **J.B. Orlin**, "A dual version of Tardo's algorithm for linear programming", 9 pages.
- 8527/A **B.M.S. van Praag**, "Household cost functions and equivalence scales", 24 pages (report 8424/E revised)
- 8528/A **S. Schim van der Loeff**, "Limited information maximum likelihood estimation of a subsystem of nonlinear equations", .. pages.
- 8529/A **B.S. van der Laan and A.S. Louter**, "On the number and the amount of damage of a passenger car traffic accidents in the Netherlands", ..pages.
- 8530/A **B.S. van der Laan and A.S. Louter**, "A statistical model for the costs of passenger car traffic accidents", .. pages.
- 8531/A **B.M.S. van Praag, J. de Leeuw and T. Kloek**, "The population-sample decomposition approach to multivariate estimation methods", 34 pages.
- 8532/A **M.E. Homan, B.M.S. van Praag and A.J.M. Hagenaars**, "Household cost functions and the value of home production in one-and two earner families", .. pages.
- 8533/B **R.J. Stroeker**, "An inequality for YFF's analogue of the Brocard angel of a plane triangle", 18 pages.

- 8534/A **J. Bouman**, "Testing nonnested linear hypothesis: A Bayesian approach based on imcomletely specified prior distributions". pages.
- 8535/A **P.M.C. de Boer, R. Harkema and B.J. van Heeswijk**, "Estimating foreign trade functions, a comment and a correction". pages.
- 8536/A **C.G.E. Boender and A.H.G. Rinnooy Kan**, "Bayesian stopping rules for multistart global optimization methods", 30 pages.
- 8537/A **A.H.G. Rinnooy Kan and G.T. Timmer**, "The multi level single linkage method for unconstrained and constrained global optimization", 14 pages.
- 8538/A **A.H.G. Rinnooy Kan**, "Probabilistic analysis of algorithms", 26 pages.
- 8539/A **A.H.G. Rinnooy Kan and G.T. Timmer**, "Stochastic global optimization methods, Part I: Clustering methods", 44 pages.
- 8540/A **A.H.G. Rinnooy Kan and G.T. Timmer**, "Stochastic global optimization methods, Part II: Multi level methods", 30 pages.

Until report 8515 this series was devided in E(conometrics), S(tatistics), M(athematics) and O(perations Research). From report 8516 on it will be

- A. Economics, Econometrics and Operations Research
- B. Mathematics
- C. Miscellaneous

