Journal of Global Optimization **13:** 241–254, 1998.

On Producing Multiple Solutions Using Repeated Trials

FRANS M. COETZEE[1] and VIRGINIA L. STONICK[2]
[1]*Siemens Corporate Research, Princeton NJ 08540, USA;* [2]*ECE Department, Carnegie Mellon University, Pittsburgh PA 15213, USA*

(Received 2 April 1996; accepted 29 December 1997)

Abstract. The number of trials that is required by an algorithm to produce a given fraction of the problem solutions with a specified level of confidence is analyzed. The analysis indicates that the number of trials required to find a large fraction of the solutions rapidly decreases as the number of solutions obtained on each trial by an algorithm increases. In applications where multiple solutions are sought, this decrease in the number of trials could potentially offset the additional computational cost of algorithms that produce multiple solutions on a single trial. The analysis framework presented is used to compare the efficiency of a homotopy algorithm to that of a Newton method by measuring both the number of trials and the number of calculations required to obtain a specified fraction of the solutions.

Key words: Exhaustive solution methods, Homotopy methods, Repeated trials

1. Introduction

Frequently a numerical solution procedure can be considered to be successful if it is capable of finding most, if not all, of the solutions to a set of nonlinear equations. A practical example is the solution of nonlinear circuit equations (Trajković et al., 1993), where all network states have to be known for optimal design. However, most standard techniques for solving systems of equations, such as Newton methods, produce a single solution from an initial point, and multiple trials are required to produce multiple solutions. In contrast, methods such as homotopy and deflation can produce multiple, and sometimes all, solutions from one initial point.

Numerical solution methods are usually compared on the basis of the computational cost of a single trial. Comparisons of algorithms based on this criterion usually favor single-solution approaches, while methods such as homotopy are frequently dismissed unless exhaustive properties can be guaranteed to offset their computational cost. Unfortunately, guarantees of exhaustive behavior are known only for some classes of functions (Drexler, 1978; Garcia and Zangwill, 1979; Morgan and Sommese, 1987), are frequently not constructive (Alexander, 1978; Diener, 1987) or are intricate to implement (Diener and Schaback, 1990). However, this paper illustrates that methods capable of producing multiple solutions on a single trial, even when not exhaustive, can perform comparably to, or show potential

computational advantages over traditional single-solution approaches, when used to find multiple solutions *using repeated trials*.

We develop a model to compare the efficiency of different algorithms at producing multiple solutions to a set of equations having a finite solution set $Q = \{x_1, x_2, \ldots, x_N\}$. Specifically, we analyze the number of trials required to obtain a fraction x of the N solutions with a given probability. Given such a model for the number of trials required, and a reasonable estimate of the relative cost of each trial for different algorithms, it is possible to choose between different algorithms.

Central to the analysis is the quantity $\mathcal{P}[X = k|N, L]$, which is the probability that *exactly* k different solutions out of the N possible solutions is obtained after a *sequence*, defined to be L successive trials. Using this quantity, the associated cumulative measure

$$\mathcal{P}[X \geqslant k|N, L] = 1 - \sum_{t=0}^{k-1} \mathcal{P}[X = t|N, L], \tag{1}$$

which is the probability that *at least* k different solutions of the N possible solutions are represented in a given sequence, can be defined. The exact calculation of these quantities depends heavily on the chosen algorithm, the problem being solved, and how repeated trials are initialized. We consider the general case of deterministic algorithms. Deterministic algorithms always produce the same solution for a given set of initial arguments to the algorithm; an example is the Newton method. It is clear that for these methods, the probability structure underlying (1) derives only from the sampling of the original argument space and the regions of convergence. This characteristic provides a rich structure for statistical modeling. Therefore, this paper focuses on describing the relationship between probabilities and regions of convergence for deterministic methods.

The general framework for estimating $\mathcal{P}[X = k|N, L]$ is formulated in Section 2. The formulation shows that this quantity can be calculated by considering equivalence classes of argument sets. However, in the general case, the combinatorial nature of the resulting problem precludes brute-force numerical calculation of the required probabilities even for small problems. Therefore, approximate theoretical bounds for some special cases, which can be applied to provide general guidelines for algorithm selection, are described in Section 2.1. The use of the derived procedures for numerical estimation of $\mathcal{P}[X = k|N, L]$ is illustrated on the six-hump camelback problem (Dixon and Szegö, 1978) in Section 3.

2. Problem formulation

A deterministic algorithm is denoted by (ψ, A, Q) where A is a set from which the starting arguments (such as the initial starting point) are chosen, Q is the set of solutions, and the solution procedure is the mapping $\psi : A \to 2^Q$, where 2^Q is the power set of Q. Denoting the cardinality of a set U by $|U|$, it follows that $|\psi(a_0)| \leqslant 1$ for all $a_0 \in A$ if the algorithm produces at most one solution for a

given initial argument, while if $|\psi(a_0)| > 1$ multiple solutions are produced on a single trial. An algorithm is globally convergent if $\psi(a_0) \neq \phi$ for all $a_0 \in A$, where ϕ denotes the empty set. For each solution $x_i \in Q$, the algorithm ψ results in an associated *basin of attraction* or *convergence region*, defined as $\Psi(x_i) = \{a \in A \mid x_i \in \psi(a)\}$. In the general case, the convergence regions of different solutions can overlap. Running the algorithm to completion once with a given $a \in A$ is called a *trial*. A *sequence* of trials results from running the algorithm L times, and is denoted by an ordered set of initial arguments $\boldsymbol{a} = (a_1, a_2, \ldots, a_L)$. Running the algorithm on a sequence implies an associated map $\psi^L : A^L \to 2^Q$ defined by $\psi^L(\boldsymbol{a}) = \cup_{i=1}^{L} \psi(a_i)$. In the rest of the paper we assume N is fixed and for notational simplicity explicit dependencies on this quantity are omitted.

In general we are unconcerned with the exact choice of arguments and only interested in the solutions that are produced. It is then convenient to group the initial arguments into equivalence classes. Since N is finite, we can number the elements of 2^Q and write $2^Q = \{ U_i \mid i = 1, 2, \ldots, 2^N\}$. Setting $\chi_{i,L} = (\psi^L)^{-1}(U_i), i = 1, 2, \ldots, 2^N$ it follows that $\chi(L) = \{\chi_{i,L}\}_{i=1}^{2^N}$ is a partition of A^L with $|\chi(L)| \leqslant 2^N$. This partition, as illustrated graphically for $\chi(1)$ in Figure 1, groups elements of A^L based on the sets of solutions which are obtained using these elements as initial arguments. We note that since Q is finite, each partition $\chi(L)$ is finite and therefore the above formulation does not require that A be either discrete or continuous.

It is clear that the partitions $\chi(L)$ for all $L > 1$ can be fully described using only knowledge of the partition $\chi(1)$. This characterization can be performed as follows: let $Z(L) = \{(i_1, i_2, \ldots, i_L) \mid i_j \in \{1, 2, \ldots, 2^N\}, j = 1, 2, \ldots, L\}$ and set $V(L) = \{\prod_{i=1}^{L} \chi_{q(i);1} \subseteq A^L \mid q \in Z(L)\}$. Then each of the elements $V_j \in V(L)$ (note $|V(L)| \leqslant 2^{NL}$) collects all sequences that pick arguments from the regions $\chi_{i;1}$ in the same order. The elements (sets) of $V(L)$ can then be grouped into N classes, based on the number of solutions produced by each sequence from the element (set) V_j. Unfortunately, finding closed form expressions for this combinatorial construction appears to be an unsolved problem.

As yet no probability structure has been defined to allow for the evaluation of (1). Since the algorithm is deterministic the probability structure derives from the way in which the initial arguments in $\boldsymbol{a}$ are chosen. In the absence of any information about the shape and location of the convergence regions, it is reasonable to assume that arguments for different trials are chosen *independently* according to the *same* distribution. We therefore assume the existence of a probability measure $\mathcal{P}$ that specifies the probability of using the initial arguments in a subset $A' \subseteq A$ for a trial. This probability measure ensures that the probability of choosing an argument on a given trial in each of 2^N equivalence classes in $\chi(1)$ is specified. Then, given the independence assumption, $\mathcal{P}$ is sufficient to calculate the induced probability measure associated with the space A^L of sequences of length L, as well as induced probability measures on the equivalence classes $\chi(L)$. Given that $\mathcal{P}$ induces all other relevant probability measures, we use the symbol $\mathcal{P}$ to denote

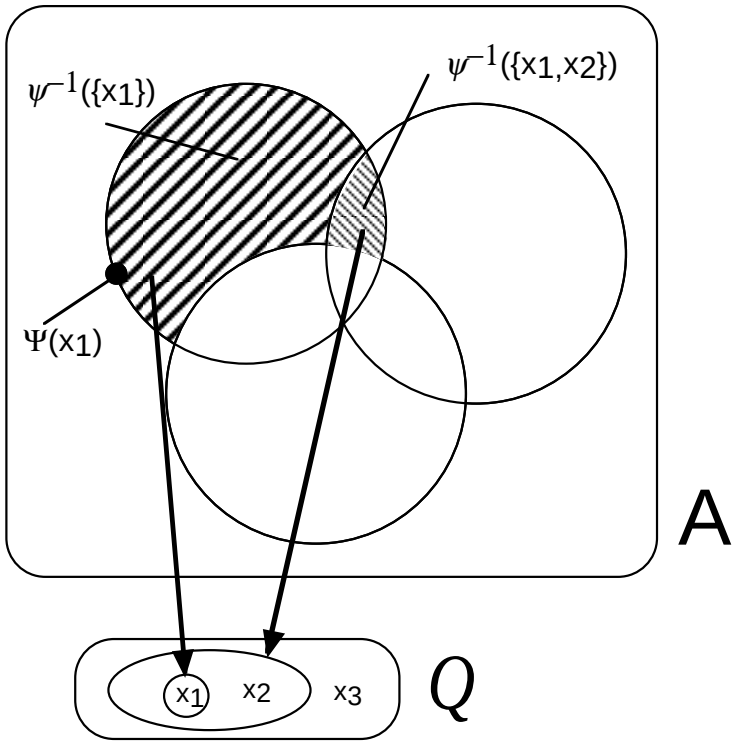

Figure 1. Problem formulation graphically illustrated for the case where there are three solutions. The circular regions of A each correspond to a convergence region for one of the solutions x_i, $i = 1, 2, 3$. The convergence regions overlap. However, a partition is defined by the sets $\chi(1) = \{\psi^{-1}(U) \mid U \in 2^Q\}$.

all probability measures throughout; in context it should be clear which measure is applicable. The probability of finding a given solution x_i on any given trial is denoted by p_i.

To calculate $\mathcal{P}[X = k|N, L]$ the sets

$$F_{(k;N,L)} = \left\{ \boldsymbol{a} = (a_1, a_2, \ldots, a_L) \in A^L \mid \left| \cup_{i=1}^{L} \psi(a_i) \right| = k \right\}, \quad k = 0, 1, 2, \ldots, N \tag{2}$$

corresponding to initial points yielding exactly k solutions in L trials have to be measured. Then

$$\mathcal{P}[X = k|N, L] = \mathcal{P}(F_{(k;N,L)}) \tag{3}$$

Equivalently, and simpler, is to calculate the probability of choosing a sequence in each of the sets $V_j \in V(L)$, and adding the probabilities of all the sets $V_j \in V(L)$ whose sequences yield exactly k solutions. Numerically the problem rapidly becomes unmanageable even for small values of N and L ($N = 5$, $L = 10$ implies up to $1.12\ 10^{15}$ sets in $V(10)$). A significant reduction in complexity results from

further grouping sets $V_j \in V(L)$ that contain sequences that are permutations of each other. Specifically, a set V_l containing a sequence $\boldsymbol{a}$ in of which n_i elements are in $\chi_{i;1}$, for $i = 1, 2, \ldots, N_0$ can be grouped with

$$\frac{L!}{n_1!n_2!\ldots n_{N_0}!} \tag{4}$$

other sets in $V(L)$. The latter approach allows for small problems to be analyzed numerically, as is done in Section 3. However, even accounting for these permutation equivalences, the computations are numerically unmanageable if the number of regions $\chi_{i;1}$ is large (above 15), and when L increases much beyond 5. It is therefore imperative that specific, simple bounds be derived for generating insight into how the efficiency of an algorithm is impacted by overlap of convergence regions. Such bounds are discussed in the next section.

2.1. ESTIMATES FOR DETERMINISTIC ALGORITHMS

To provide estimates of the probabilities of finding different numbers of solutions we consider three cases. In the first case ψ produces at most one solution for a given initial set of arguments. In the second case, the general problem where the basins of attraction overlap partially is discussed. Finally, we consider the case where ψ will always produce the solution x_i when x_j is produced, and *vice versa*.

When each initial point produces exactly one solution, the basins of attraction are disjoint and form a partition of the set $\cup_{i=1}^{N} \Psi(x_i)$. If it is assumed that all basins of attraction have roughly the same measure m and the argument space is sampled uniformly, the probabilities p_i can be assumed to be equal. Equivalently, it can be assumed that the basins of attraction differ in measure, but are sampled such that the probabilities p_i are equal, i.e. the small basins of attraction are sampled more heavily. If the algorithm is globally convergent the basins of attraction cover A and hence the assumptions above lead to $p_i = 1/N$ (if the algorithm fails to produce a solution for $A' \subset A$, where $\mathcal{P}(A') = \alpha$, the probabilities are decreased to $p_i = (1 - \alpha)/N$).

In the second case, the basins of attraction of the solutions overlap, but are not the same, i.e. $\Psi(x_i) \cap \Psi(x_j) \neq \phi$, $\Psi(x_i) \neq \Psi(x_j)$. To illustrate the effect of this partial overlap, consider a system with two solutions $Q = \{x_1, x_2\}$, and where the regions of attraction partially overlap. The problem is symmetric in x_1 and x_2. For two trials ($L = 2$) the following set of events is possible:

$$\begin{array}{ccc} (\{x_1\}, \{x_1\}) & (\{x_1\}, \{x_2\}) & (\{x_1\}, \{x_1, x_2\}) \\ (\{x_2\}, \{x_1\}) & (\{x_2\}, \{x_2\}) & (\{x_2\}, \{x_1, x_2\}) \\ (\{x_1, x_2\}, \{x_1\}) & (\{x_1, x_2\}, \{x_2\}) & (\{x_1, x_2\}, \{x_1, x_2\}) \end{array} \tag{5}$$

Note that the probability of finding x_1 on a single trial is now given by $p_1 = \mathcal{P}(\{x_1\}) + \mathcal{P}(\{x_1, x_2\}) \geqslant \mathcal{P}(\{x_1\})$. Applying symmetry arguments and assuming

that $\mathcal{P}(\{x_1, x_2\}) = \lambda p_1$, it is straightforward to show that the probability of finding both solutions in two trials is given by

$$1 - 2\left(\frac{1-\lambda}{2-\lambda}\right)^2 \tag{6}$$

Here λ is a real factor proportional to the overlap of the regions of convergence and measures the probability of obtaining one solution given that the other solution has been found. When $\lambda = 1$, the regions overlap fully; when $\lambda = 0$, the regions are disjoint. Therefore, as λ is varied, various problems are generated intermediate to the case considered above, and the case where all solutions have the same convergence region. Based on Expression (6), it is clear that the effect of increasing overlap is to increase the probability of finding more solutions in a given period of time. This increase results since the overlap increases the probability p_i associated with a given solution, since it is no longer required that $\sum_i p_i = 1$.

As N increases, a full accounting of joint probabilities of all combinations of solutions defies analysis. Not only does the number of probabilities that has to be specified increase (even assuming symmetry analogous to the above example a total of N probabilities have to be specified), but at issue is exactly how the convergence regions overlap. For example, for the symmetric case as regions increasingly overlap ($\lambda = 1$ in the problem above), a single convergence region exists and the method is exhaustive. A more reasonable occurrence is to consider problems where groups of solutions have significant overlap.

This leads us to the final case, where an algorithm ψ always produces the solution x_i when x_j is produced, and vice versa. For this case it is possible to define equivalence classes of solutions,

$$[x_i] = \{x_j \in Q \mid \Psi(x_i) = \Psi(x_j)\}. \tag{7}$$

Each equivalence class has an associated probability $\mathcal{P}([x_i]) = p_i$ and a basin of attraction $\Psi([x_i]) = \Psi(x_i)$. Since equivalence classes are by definition disjoint, this case corresponds to solving a problem where Q consists of equivalence classes, and has fewer solutions than the original problem. For example, assume that the original set Q can be partitioned into a set of equivalence classes, such that each of the equivalence classes contains exactly r of the original N solutions. If the basins of attraction of the solutions are assumed equal then $\mathcal{P}([x_i]) \simeq r/N = 1/(N/r)$. The problem then corresponds to the first special case described above but having only N/r solutions in Q.

Based on the three cases discussed above, it is reasonable to view the general case where convergence regions overlap as a problem that is intermediate to two cases. At one extreme the algorithm produces one solution on each trial and the problem has N non-overlapping convergence regions. At the other extreme the algorithm produces groups of solutions at a time, which can be considered as solving a problem with fewer solutions with non-overlapping convergence regions. In the next section we provide a common statistical model for these two cases.

2.2. NON-OVERLAPPING EQUAL-PROBABILITY CONVERGENCE REGIONS

Consider an algorithm where each trial produces one solution x_i, and for which only one representative in each convergence region is kept. A sequence can then be represented by the solutions $(x_{i_1}, x_{i_2}, \ldots, x_{i_L})$ which result. Each of these samples has an associated probability depending on the size of the convergence region of the solutions present in the combination. Only the equal probability case is readily analyzable, in which case the probability $\mathcal{P}[X = k|N, L]$ is given by the fraction of sequences satisfying the constraint which defines the corresponding event.

Consider the set of strings $(x_{i_1}, x_{i_2}, \ldots, x_{i_L})$ where $x_{i_j} \in Q$. The number of strings satisfying the constraint that there are exactly k different elements present from an alphabet of N solutions in the string of length L is denoted by $\eta(k; N, L)$. Given N elements in the alphabet, there are ${}_N C_k$ possible subsets of k different elements. Using any k such elements as a new alphabet, there are a total of k^L strings which can be constructed. From the definition of $\eta(k; N, L)$, it follows that of these k^L strings in the new alphabet, $\eta(j; k, L)$ have exactly j elements present where $j \in \{1, 2, \ldots, k\}$. Hence, there are $k^L - \sum_{i=1}^{k-1} \eta(i; k, L)$ strings that are constructed with the new alphabet that have exactly k different elements represented. Therefore, the total number of strings that can be constructed with k different solutions of length L, is given by

$$\eta(k; N, L) = {}_N C_k \left[k^L - \sum_{i=1}^{k-1} \eta(i; k, L) \right] \tag{8}$$

Using the fact that $\eta(1; N, L) = N$, the above equation specifies an iterative procedure for calculating $\eta(k; N, L)$. Since there is a total of N^L possible samples, it follows that

$$\mathcal{P}[X = k|N, L] = \frac{\eta(k; N, L)}{N^L} \tag{9}$$

and it is possible to write the following difference equation for the probabilities:

$$\mathcal{P}[X = k|N, L] = {}_N C_k \left(\frac{k}{N} \right)^L \left[1 - \sum_{i=1}^{k-1} \mathcal{P}[X = i|k, L] \right] \tag{10}$$

To bound this expression, consider the specific case of calculating $\mathcal{P}[X = N|N, N]$, i.e. the probability of finding all N solutions in the minimum number of successive trials. Since the order in which the solutions are produced is irrelevant it follows that all permutations $\sigma \in S^N$, the set of permutations of N elements, should be considered:

$$\begin{aligned} \mathcal{P}[X = N|N, N] &= \sum_{\sigma \in S^N} \mathcal{P}(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(N)}) \\ &= \left(\prod_{i=1}^{N} p(x_{\sigma(i)}) \right) \left(\sum_{\sigma \in S^N} 1 \right) = N! \prod_{i=1}^{N} p_i \end{aligned} \tag{11}$$

This expression can be bounded above and below:

$$N!\left(\min_i\{p_i\}\right)^N \leqslant \mathcal{P}[X = N|N, N] \leqslant N!\left(\frac{1}{N}\right)^N \tag{12}$$

Using Stirling's approximation to the factorial, the upper bound can be written as

$$N!\left(\frac{1}{N}\right)^N \simeq \left(\frac{N}{e}\right)^N \left(\frac{1}{N}\right)^N = \left(\frac{1}{e}\right)^N. \tag{13}$$

2.3. DISCUSSION

Using the above derived bounds, it is now possible to make some general statements about the desirability of algorithms that produce multiple solutions on a trial. Consider first the case where the solutions can be grouped into equivalence classes each containing approximately r solutions and with equal probability. The probability of producing all N solutions in the minimum number of trials is given by $\mathcal{P}[X = N/r|N/r, N/r]$. A graph of the upper bound (12) of this quantity *versus* N is shown in Figure 2. On this graph the curve for $r = 1$ corresponds to a method producing only one solution on each trial. As expected, the probability of finding all solutions in the minimum number of trials becomes vanishingly small as the number of solutions to a problem increases. Expression (13) describes the asymptote of the curve. It is clear that any method capable of producing on average two solutions on the same trial will perform dramatically better at producing solutions using the minimum number of trials than a method producing only one solution on a trial. The exponential bound in (13) shows that even for relatively small N a substantial computational cost increase of a multiple solution procedure over a single solution procedure could be offset by the smaller number of trials required by the former to find a large subset of the solutions.

The second quantity of interest is the quantity $\mathcal{P}[X \geqslant N|N, L]$, which is the probability that all possible solutions will be obtained in L trials. This quantity is shown in Figure 3 for different N. Figure 4 shows how many trials are needed to ensure that with probability 0.95 a specified fraction x of N solutions have been found; it shows L such that $\mathcal{P}[X \geqslant xN|N, L] \geqslant 0.95$. From both figures the value of producing multiple solutions is clear from the sharp decrease in the number of trials that results to reach a given fraction of the solutions with a specified confidence level. Furthermore, as the probability threshold is raised this decrease becomes increasingly pronounced.

Based on these results it is clear that significantly fewer trials are generally required for a given level of performance, if multiple solutions are obtained on a trial. This decrease in the number of trials can potentially compensate for a higher computational cost per trial.

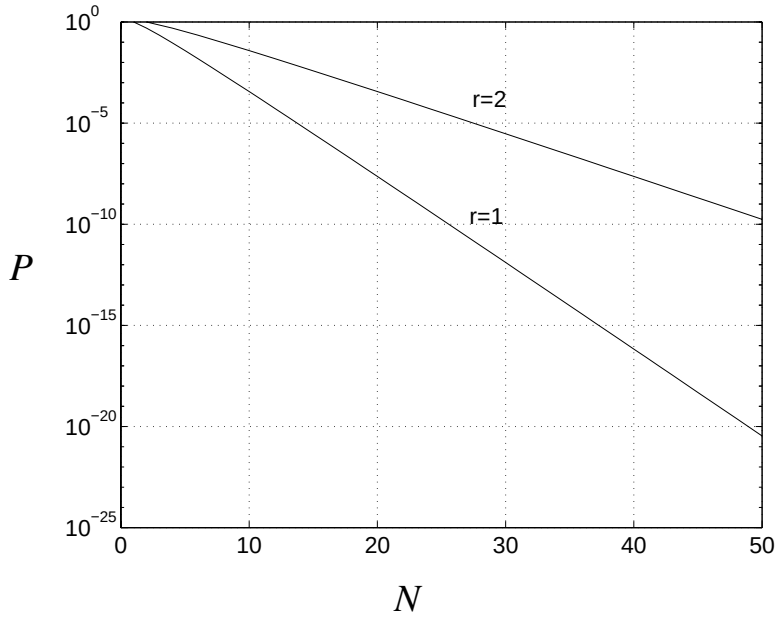

Figure 2. Upper bound on the probability P of finding all N solutions using only N/r trials, where on average r solutions are produced on a trial.

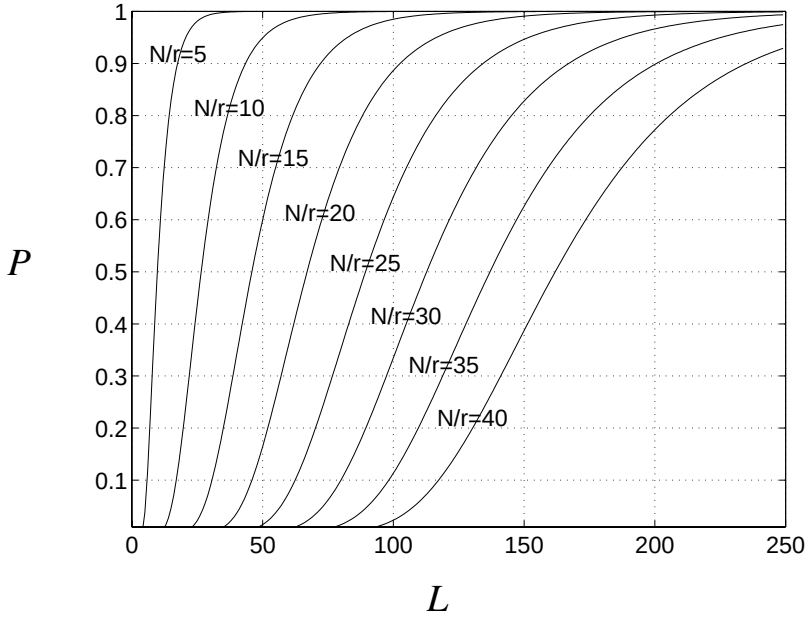

Figure 3. Probability P of finding all N solutions in L trials as a function of L, for different values of r, the number of solutions found on average on a single trial.

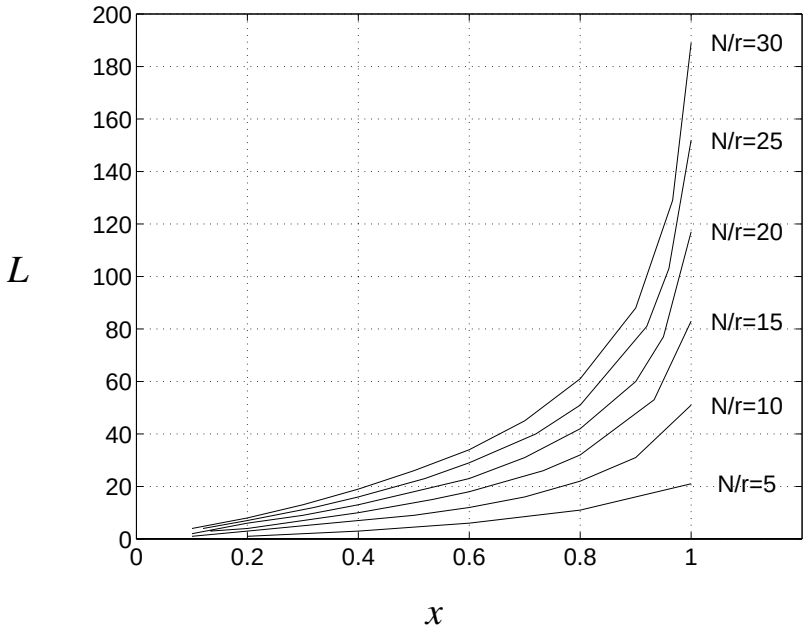

Figure 4. Minimum number of trials L needed to obtain a fraction x of N solutions with probability 0.95.

3. Example test problem

The six-hump camelback problem (Dixon and Szegö, 1978) was used as a test problem. The objective is to find the fifteen critical points of the equation

$$f(\boldsymbol{x}) = x_1^6/3 - 2.1x_1^4 + 4x_1^2 + x_1x_2 - 4x_2^2 + 4x_2^4 \tag{14}$$

in the region $[-2.0, 2.0]^2$. Three algorithms were compared for solving this problem. The first algorithm is a damped Newton method, where a new estimate x^+ of the solution is generated by

$$x^+ = x^- - \frac{1}{4}[D_x^2 f(x^-)]^{-1} D_x f(x^-) \tag{15}$$

from the current solution estimate x^-. Damping was introduced to ensure reasonable convergence regions for all solutions from the initial points that were considered. In addition, the problem was solved using a recently introduced and generally applicable two-stage sequential homotopy algorithm (Coetzee, 1995; Coetzee and Stonick, 1995). The two-stage homotopy significantly increases the number of solutions a given homotopy can obtain from a single initial point, but in general is not exhaustive. These two algorithms are compared to the ideal globally convergent single-solution algorithm, with each solution appearing with equal probability. For this case, the theoretical results obtained in Section 2.1 hold.

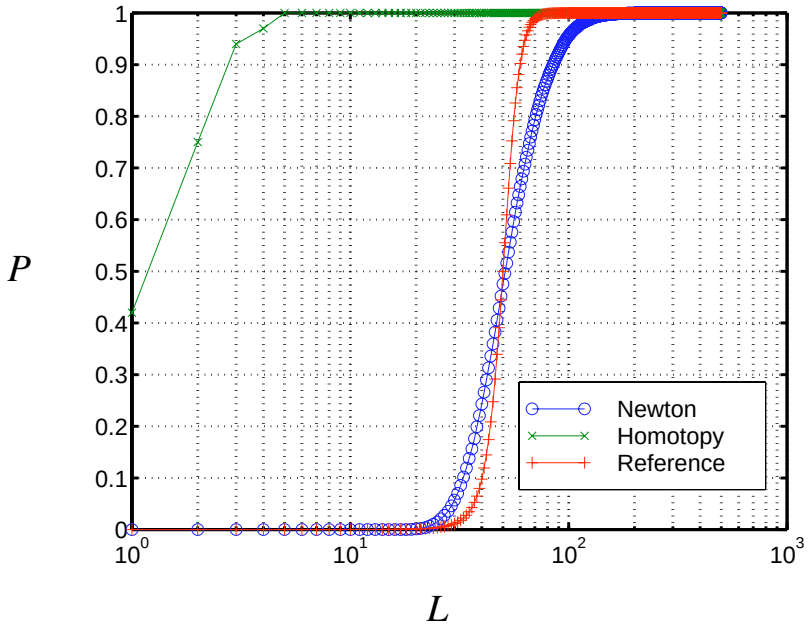

Figure 5. Probability P of finding all fifteen solutions to the camelback problem as a function of the number of trials L. The reference algorithm is an algorithm which produces one solution on a trial, where all convergence regions have equal probability.

For the first two algorithms repeated trials were performed on a grid of 324 initial points. On all of the trials a list was created containing each initial point and the solutions that resulted from the trial. Using these results, a program was written to numerically calculate the estimates of $\mathcal{P}[X = k|N, L]$ and $\mathcal{P}[X \geqslant k|N, L]$ as described in Section 2.1. These calculations are numerically feasible due to the fact that both algorithms have only a few regions represented in $\chi(1)$, and each solution has a high probability of occurrence. For the homotopy algorithm, this property results from the fact that convergence regions overlap significantly (on average 11 solutions were found on a trial) while for the Newton method this property results since exactly one solution can be obtained on a trial.

The numerical calculations yield the relationships shown in Figure 5 and Figure 6. Figure 5 shows the probability of finding all solutions as the number of trials L increases. As expected, the probability of finding all solutions in a few trials is much greater using the homotopy approach, with a given confidence level being reached typically in an order of magnitude fewer trials. The damped Newton method typically finds more solutions than the idealized algorithm for small values of L due to the slightly larger convergence regions of some solutions. However, as the fraction of solutions required approaches one, the difficulty in finding the solutions with the smaller convergence regions results in more trials than for the

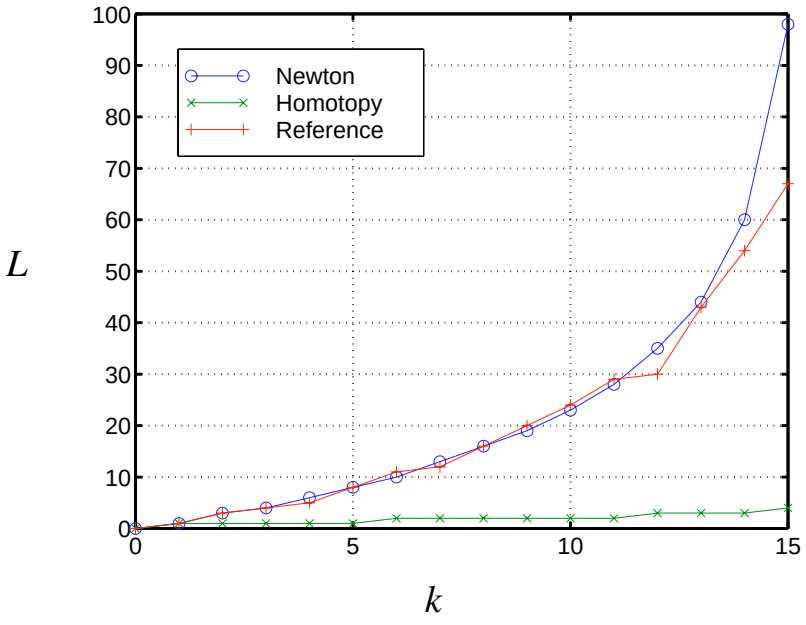

Figure 6. Minimum number of trials L needed for 0.95 probability that k of the fifteen solutions to the camelback function are obtained.

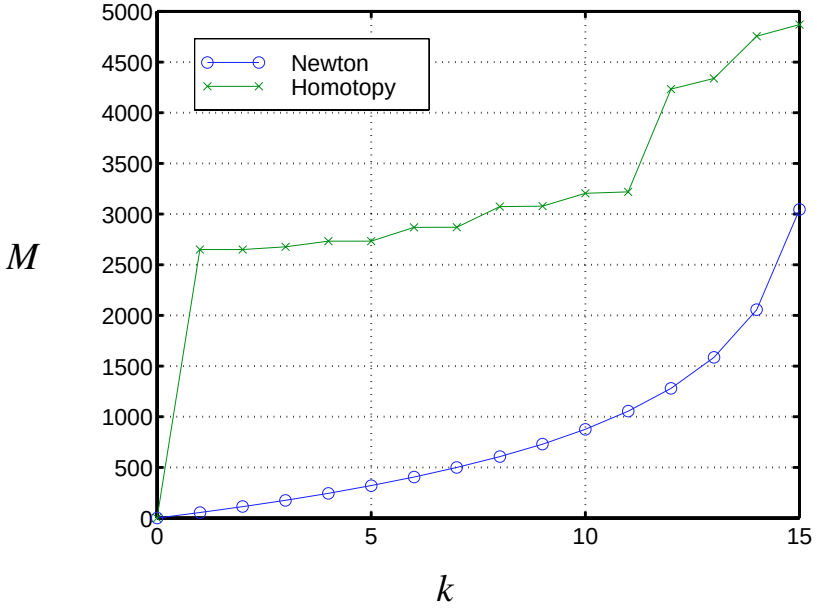

Figure 7. Number M of equivalent Newton steps required to obtain k of the solutions. Each Newton step requires solving a linear system of equations.

idealized test case. Figure 6 shows the minimum number of trials required to obtain k of the solutions with 0.95 probability. It is clear that the large number of solutions on a given homotopy path radically reduces the number of trials required to find any fraction of the solutions with reasonable confidence.

To illustrate the impact of the fewer number of trials on the total computational cost, the computational expenditure of the homotopy algorithm is compared to that of the Newton algorithm. At each step along the continuation path, the homotopy algorithm iterates using Newton's algorithm, which involves solving a system of two linear equations. Figure 7 shows the average number of Newton iterations M required by each method to find k of the solutions. It was found that the damped Newton method required on average 55 iterations to converge to the required accuracy. The homotopy algorithm averaged 2650 Newton iterations on each trial, which is approximately 53 times as many iterations per trial as that of the Newton method. However, the average cost to find all solutions are 4870 Newton iterations for the homotopy method, and 3050 for the damped Newton method, which implies a much smaller overall relative computational complexity of 1.60.

4. Conclusions

Using an algorithm that produces multiple solutions on a single trial, rather than a method that yields only one solution on a trial, results in a sharp decrease in the number of trials to find a specified fraction of the solutions. When large numbers of solutions are required, this decrease in the number of trials could compensate for the additional computational cost of producing multiple solutions on a single trial. A guaranteed exhaustive method clearly represents one extreme since almost any reasonable amount of computation will result in the superiority of such an algorithm over other approaches. However, in the absence of such a method, it is valuable to consider extending existing algorithms to obtain more solutions on a single trial, if the computational cost is reasonable. Incorporating deflation techniques, where solutions are removed from the solution set as they are found, seems especially useful. Sequential homotopy methods are especially amenable to the latter approach.

Acknowledgments

The authors would like to thank Prof. J.M.F. Moura for an interesting discussion, and an anonymous reviewer for suggestions that substantially improved the rigor of the presentation. This work was supported in part by NSF grant MIP-9157221.

References

Alexander, J.C. (1978), The topological theory of an embedding method, in H. Wacker (ed.), *Continuation Methods* (pp. 37–68), Academic Press.

Coetzee, F.M. (1995), Homotopy approaches for the analysis and solution of neural network and other nonlinear systems of equations, Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA.

Coetzee, F.M. and Stonick, V.L. (1995), Sequential homotopy-based computation of multiple solutions to nonlinear equations, *Proceedings ICASSP 1995*, IEEE.

Diener, I. (1987), On the global convergence of path following methods to determine all solutions to a system of nonlinear equations, *Mathematical Programming* 39: 181–188.

Diener, I. and Schaback, R. (1990), An extended continuous Newton method, *Journal of Optimization Theory and Applications* 67(1): 57–77.

Dixon, L.C.W. and Szegö, G.P. (1978), The global optimization problem: An introduction, in L.C.W. Dixon and G.P. Szegö (eds), *Towards Global Optimization 2* (pp. 1–15), North-Holland, Amsterdam.

Drexler, F.J. (1978), A homotopy-method for the calculation of all zero-dimensional polynomial ideals, in H. Wacker (ed.), *Continuation Methods* (pp. 69–93), Academic Press.

Garcia, C.B. and Zangwill, W.I. (1979), Determining all solutions to certain systems of nonlinear equations, *Mathematics of Operations Research* 4(1): 1–14.

Morgan, A. and Sommese, A. (1987), Computing all solutions to polynomial systems using homotopy continuation, *Applied Mathematics and Computation* 24: 115–138.

Trajković, L., Melville, R.C., Fang, S.-C. and Watson, L.T. (1993), Artificial parameter homotopy methods for the d.c. operating point problem, *IEEE Trans. on CAD* 12: 861–877.