Journal of Global Optimization (2005) 31: 567–578
DOI 10.1007/s10898-004-9966-0

On the Investigation of Stochastic Global Optimization Algorithms

BILL BARITOMPA and ELIGIUS M.T. HENDRIX
Department of Mathematics and Computing, University of Canterbury, Christchurch, New Zealand (e-mail: b.baritompa@math.canterbury.ac.nz)

Abstract. This discussion paper for the SGO 2001 Workshop considers the process of investigating stochastic global optimization algorithms. It outlines a general plan for the systematic study of their behavior. It raises questions about performance criteria, characteristics of test cases and classification of algorithms.

1. Introduction

This paper is a discussion document which preceded the SGO 2001 Workshop. We are interested in properties of Global Optimization (GO) algorithms and their interrelation with the mathematical structure of GO problems. The goal is to understand which methods are best suited to which type of practical optimization problem. The purpose of this paper is to ask questions and engender discussion. It is hoped that some of the aspects discussed in this paper will be further investigated at the workshop.

An outline of discussion topics is: Section 2 – general overview; section 3 – performance criteria; section 4 – characteristics of test cases; section 5 – classification of algorithms; and section 6 – summary and discussion points.

2. General Overview

We are interested in studying the behavior of algorithms in a systematic way. To do this we need a plan and agreed upon factors.

2.1. PLAN

1. Formulation of performance criteria.
2. Description of the algorithm(s) under investigation.
3. Selection of appropriate algorithm parameters.
4. Production of test functions (instances, special cases) corresponding to certain landscape *structures or characteristics.*
5. Analysis of its theoretical performance, or empirical testing.

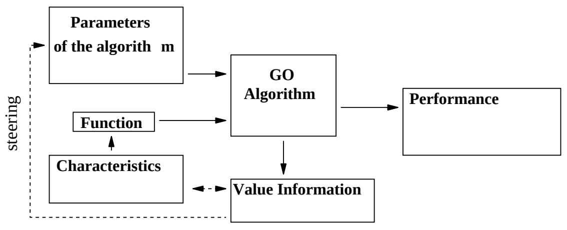

Figure 1. Aspects in investigating stochastic global optimization algorithms.

Figure 1 depicts some relevant aspects.

2.2. EXAMPLE

1. Performance criterion: The probability P of "hitting" a global optimum.
2. The algorithm: Pure Random Search on a compact robust set where trial points are generated unifomly from the set.
3. Parameters: N the number of trial points.
4. Test functions: Make functions with specific value for $\frac{V(B_\delta(S^*))}{V(X)}$, the relative volume of the δ-neighborhood of all global minimum points.
5. Analysis:

$$P = 1 - \left(1 - \frac{V(B_\delta(S^*))}{V(X)}\right)^N \tag{1}$$

Apparently the assumed target of a user is to reach one global minimum point. The search is assumed successful when the record value hits a δ-neighborhood of a global minimum point and the researcher has formulated as a performance criterion the probability of success. Alternatively, the success could be defined as being in an ϵ-level set from the global minimum value f^*, where the probability of success becomes:

$$P = 1 - (1 - \mu(f^* + \epsilon))^N \tag{2}$$

depending on the relative size μ of the $\epsilon + f^*$-level set as a characteristic of the problem to be solved.

The main theme is that all aspects must be considered together, as is now detailed.

2.3. PERFORMANCE CRITERIA

In section 3, we discuss performance criteria that are used in Global optimization. We do not expect to find *one* universal criterion, but are interested in a variety and the interrelationships between them. Moreover, we describe a tool which we call the *Performance Graph.*

2.4. CHARACTERISTICS

Experimentally an algorithm is often run over several test functions and its performance compared to other algorithms and/or other parameter settings. To understand behavior we need to study the relationships to characteristics of landscapes of test functions. The main question is how to define appropriate characteristics. We will discuss some ideas which appear in the literature in section 4. The main idea is that relevant characteristics depend on the type of algorithm as well as on the performance measure. In the previous example only the relative size of the sought for region matters, and characteristics such as the shape of regions of attraction, the form of the finite level sets, and barriers in the landscape do not matter.

2.5. ALGORITHMS

An interesting discussion and classification of algorithms can be found in [12]. According to [5], one can roughly distinguish two major approaches:

- Deterministic methods which guarantee to approach the global optimum and require a certain mathematical structure.
- Stochastic methods which are based on the random generation of feasible trial points and nonlinear local optimization procedures.

In this discussion paper, the focus is on stochastic optimization procedures, although many concepts are more generally applicable. General stochastic algorithms require no structural information about the problem. One of the most generic descriptions is that of Törn and Zilinskas [12] in which x_k are chosen stochastically in sequence according to the general equation:

$$x_{k+1} = Alg(x_k, x_{k-1}, \dots, x_0, \xi) \tag{3}$$

where ξ is a random variable.

We do not think a generic stochastic algorithm is directly useful for practical problems. However, one can adapt algorithms to make use of structural information. Moreover, one should notice, that even if structural information is not available, other so-called *value information*, becomes available when running algorithms. For example: the number of local optima found thus far, the average number of function evaluations necessary for one local search, the best function value found, and the behavior of the local search, etc. Such indicators can be measured empirically and can be used to get insight into what factors determine the behavior of a particular algorithm and perhaps can be used to improve the performance of an algorithm.

Can we come to a classification of stochastic global optimization algorithms? We look at this further in section 5.

3. Performance criteria

There are two questions to address.

- Effectiveness: does the algorithm find what we want?
- Efficiency: what are the computational costs?

The GO literature looks at each separately. We suggest a good criterion needs to combine both. We suggest the performance graph for studying and picturing the trade-off between the two main criteria.

Over the last 10 years, starting from the early works of Dixon and Szegö [3] and following publications in the Journal of Global Optimization, the main study of effectiveness is to that show when effort goes to infinity, a point is generated in a neighborhood of global minimum point with probability one. Analytically this main property can be derived whenever a designed algorithm is allowed to sample everywhere in the feasible area with a certain probability. In general with stochastic methods, we need many repetitions to measure performance criteria, in order to average out the stochasticity.

Empirically, by means of test functions, one can measure how many times the global optimum has been reached by a certain algorithm. In this context, *simulation*, *experiments* or *computational results* can explore performance.

Efficiency can also be measured empirically. The expected number of function evaluations necessary to reach a particular level of convergence, is taken. In the literature experimental results are presented as tables where several algorithms (or one algorithm with several parameter settings) are run over some test functions and the required number of function evaluations to reach the global optimum is reported.

3.1. TARGETS

What are other good ways to define the performance of an algorithm? Focusing on effectiveness there are several targets a user of the algorithm could have in mind:

1. To discover all global minimum points. This of course can only be realized when the number of global minimum points is finite.
2. To detect at least one global optimal point.
3. To find a solution with an “acceptably low” function value.
4. To produce a uniform covering: This idea as introduced by Klepper and Hendrix [6, 10], is that the final population of a population based algorithm should resemble a sample from a uniform distribution over a level set with a predefined level. The practical relevance is due to identification problems in parameter estimation. Simple performance indicators as to how well this has been achieved are difficult to con-

struct. Usually one partitions the final region in subsets and sums the deviations from the expected number of points in each partition set.

The first and second targets are typical satisfaction targets. Namely, was the search successful or not? What are good *measures of success*? In the older literature, often convergence was used. If we want to make results comparable, we need to be more explicit in our definitions of success. We need to address the following questions:

- What is considered "low", a predefined level, a percentile of the function range, or a δ-neighborhood of the global minimum point(s)?
- When the record of the search process hits a "low" level set, did it "see" the minimum only once or repeatedly?
- Does the process converge to a minimum point (or in population algorithms around several minimum points) to give the user a feeling this is really a minimum point?

3.2. EFFICIENCY

Globally, efficiency is defined as the effort the algorithm needs to be successful. A usual indicator in stochastic algorithms is the (expected) number of function evaluations necessary to reach the optimum. This indicator depends on many factors such as the shape of the test function and the termination criteria used. By making more and more assumptions on the behavior of the algorithm, one can come up with the complete distribution of number of function evaluations necessary to reach the optimum.

An alternative efficiency indicator is the Success Rate [7] defined as the probability that the next iterate is an improvement on the record value found thus far. Its relevance to convergence speed was analyzed by Zabinsky and Smith [13] and Baritompa et al. [1], who showed that a fixed success rate of an effective algorithm (in the sense of uniform covering) gives an algorithm with the expected number of function evaluations growing polynomially with the dimension of the problem. However, the empirical measurements can only be established in the limit when such an algorithm stabilizes, and only for specifically designed test cases.

3.3. PERFORMANCE GRAPH

Analytical studies often aim at showing the algorithm is successful in the limit. In no realistic situation is a user going to use infinite effort! The question is really how to measure the performance for cases where the user has a finite amount of time (in [4] called a budget) to obtain a solution. We introduce the concept of a Performance Graph. If success is Boolean, the graph plots the probability of success (depending on the success

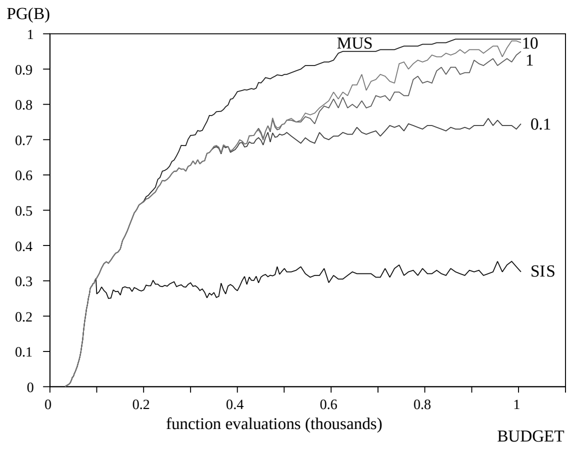

Figure 2. Probability of reaching the optimum for the Shekel-5 function and several algorithms given a number of function evaluations.

indicator) versus the effort required. An example [4] is given in Figure 2; an estimate is given of the probability of success,[1] based on running algorithms over many repetitions for a given amount of function evaluations. Note quite a few aspects of a method can be seen from the performance graph. For example the method called SIS become marginally ineffective. The method called MUS is better than method 10 for low effort.

In a non-boolean case like target three, the performance graph plots the average of observed values of the goal versus effort. An example is given in Figure 3. With sufficient assumptions, a theoretical analysis of a stochastic method could compute the exact expected value. In that case the performance graph would plot the expected value versus effort.

3.4. RELATION BETWEEN CRITERIA

The performance graph emphasizes the relationship between success and effort. In other studies the focus is on the distribution of effort for a specified degree of success (or convergence). How are those two concepts related? In particular if success is defined as seeing a point in the level-set $S(y)$, as we move the level y upwards, the graph giving the probability of success slowly moves upwards. How does this relate to the performance graph? What is the relation with the graph which measures the expected (or distribution of the) record values? Can a performance graph be derived analytically from a given success rate (fixed rate of improvement)?

[1]Success in this example was defined as the record being in a neighborhood of the one global minimum point.

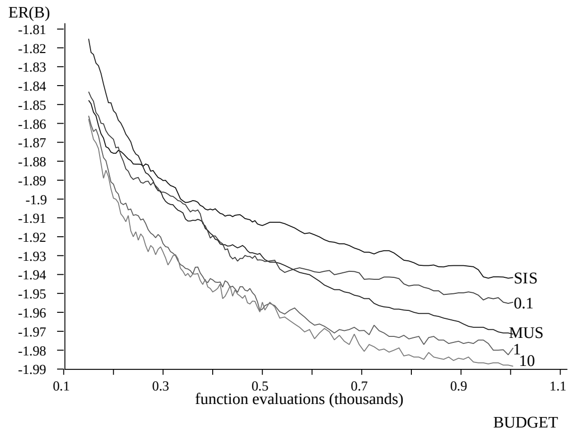

Figure 3. Average record value reached by 5 different algorithms for a given number of function evaluations for the Rastrigin test function.

3.5. COMPARISON OF ALGORITHMS

When comparing algorithms, a specific algorithm is *dominated* if there is another algorithm which performs better (e.g. has a higher performance graph) in all possible cases under consideration. Usually however, one algorithm runs better on some cases and another on other cases (e.g. in Figures 2 and 3, where algorithm *MUS* does best on the Shekel-5 function, but there are better ones on the Rastrigin function).

So basically the performance of algorithms can be compared on the same test function, or preferably for many test functions with the same characteristic, where that characteristic is the only parameter that matters for the performance of the compared algorithm. As we have seen in section 4, it may be very hard to find out such determination characteristics. The following principles can be useful:

- Comparability: when comparing several algorithms they should all make use of the same type of (structural) information (same stopping criteria, accuracies etc).
- Simple references: it is wise to include in the comparison simple benchmark algorithms such as Pure Random Search, Multi-start and Pure Adaptive Search in order not to let analysis of the outcomes get lost in parameter tuning and complicated schemes.

Often in the literature we see algorithms applied for solving "practical" problems. If we are comparing algorithms for a practical problem, we should keep in mind this is only one problem and up to now, nobody has defined what is a representative practical problem.

4. Characteristics of Test Cases

As empirical results depend on the test functions under consideration, we need to look carefully at the structure of the functions and their corresponding landscapes. Which factors determine the performance? As in design of experiments, one should construct extreme test cases (best or worst) to aid investigation of suspected relations.

4.1. EUCLIDEAN LANDSCAPE IS MISLEADING

Intuition gleaned from thinking about graphs of real valued functions of two variables, whose graphs are surfaces resembling landscapes found in our "Euclidean" world, is of limited value. The language of hills and valleys, needs to be extended to higher dimensions. We should try to think more abstractly and see the landscape from the perspective of the algorithm and its assumed neighborhood structure.

For pure random search, the next generated point does not depend on the current iterate and therefore we must consider all points to be in the neighborhood of the current iterate. In this perspective local non-global optima do not exist. We should redefine the neighborhood structure as all points which can be reached (with non-zero probability) from a certain point given the operation of the algorithm given by (3).

This requires a new topological structure. If we could define it, we would get to the heart of what are relevant characteristics to measure. It is a challenge to see algorithms and spaces from this perspective. A particularly challenging question is how to define the relevant geometry to study population based algorithms.

4.2. SPECIFIC CONSIDERATIONS

It will be important to vary the test cases systematically between the extreme cases, in order to understand how algorithms behave. In an experimental setting, depending on what one measures, one tries to design experiments which yield as much information as possible. However, to derive analytical results, it is not uncommon to make highly specific assumptions which make the analysis tractable. For example:

- To study the limit behavior for a unique optimum, the spherical or conical form of the function is assumed.
- To illustrate complexity, the concave quadratic problem over a hypercube has been used. There the number of local optima (2^n) increases exponentially in the dimension n.
- In nonlinear optimization the condition number of the Hessian at the minimum point is often specified, as it influences efficiency of local nonlinear optimization algorithms.

When studying cases we should keep in mind that in the GO literature the following types of problems have been investigated.

- Black box case: in this case it is assumed that nothing is known about the function to be optimized. Often the feasible set is defined as a box, but information about the objective function can only be obtained by evaluating the function at feasible points. This is also called *the oracle case*.
- Grey box case: something is known about the function, but the explicit form is not necessarily given. We may have a lower bound on the function value or the number of global and/or local optima. As have proven useful for deterministic methods, we may have structural information such as: the function is concave, a Lipschitz constant is known, a d.c.-decomposition is known. Stochastic methods often do not use this type of information, but it may be used to derive analytical or experimental results.
- White box case: explicit analytical expressions of the problem to be solved are assumed to be available. Specifically interval arithmetic algorithms require this point of view on the problem to be solved.

When looking at the structure of the instances for which we study the behavior of the algorithm we should keep two things in mind:

- In experiments the researcher can try to influence the characteristics of the test cases such that the effect on what is measured is as big as possible. Note that the experimentalist knows the structure in advance, but the algorithm does not.
- The algorithm can try to generate information which tells it about the landscape of the problems.

4.3. COMPLICATING FACTORS

A difficulty in the analysis of a GO algorithm in the multi-extremal case is, that everything seems to influence behavior:

- the orientation of components of lower level sets with respect to each other determine how iterates can jump from one place to the other.
- the number of local optima up in the "hills" determine how algorithms may get stuck in local optima.
- the difference between the global minimum and the next lowest minimum affects the possibility to detect the global minimum point.
- the steepness around minimum points, valleys, creeks etc. which determine the landscape determine the success.

However, we stress, as shown in our first example, the *characteristics* which are important for the behavior depend on the *type of algorithm* and the *performance criteria* that describe the behavior.

So in every analytical and experimental investigation of an algorithm we should be concerned with the type of characteristics which matter for the algorithm and performance criterion under consideration. Can we classify this? We suggest for various classes of methods the following:

- Multi-start type algorithms: size of regions of attractions of optima, number of local and global optima.
- Clustering algorithms: the above together with the shape of regions of attraction.
- Population based algorithm with focus on success rate: number, orientation (Hessian) of components of lower (limit) level sets.
- Hit and Run, MC type of algorithms with focus on expected number of function evaluations: barriers, creeks and rivers, location, orientation and number of saddle points.

4.4. USEFUL INFORMATION

If we want to look for weaknesses and strengths of algorithms we should consciously look for extreme cases with respect to these characteristics. It may be hard to measure characteristics from given test cases. Various graphical techniques predict the success of some random schemes. The graph of the relative measure of the improving region versus function values predicts the success of some random schemes. In [4] a characterizing graphcan be found that determines how successful it is to do more or less localsearches. A clear drawback of this characterization is the difficulty of finding the complete graph describing the relevant structure of the landscape.

From the perspective of designing algorithms, running them empirically may *generate information* about the landscape of the problem to be solved. A list of information one could measure during running a stochastic GO algorithm on a black box case from [5] is useful:

- Graphical information on the decision space.
- Current function value.
- Best function value found so far (record).
- Number of evaluations in the current local phase.
- Number of optima found.
- Number of times each detected minimum point is found.
- Estimates of the time of one function evaluation.
- Estimates of the number of function evaluations for one local search.

- Estimates of the likelihood[2] the optimum has been reached.

5. Classification of Algorithms

Stochastic methods are understood to contain some stochastic elements. Either the outcome of the method is a random variable or the objective function itself is considered a realization of a stochastic process. For an overview on stochastic methods we refer to Törn and Zilinskas [12] and Boender and Romeijn [2]. Apart from the continuity of *f*, the methods in general require no structure on the optimization problem and therefore are generally applicable. If we want to compare algorithms, then it is useful to do this from the perspective of the same information being used and the same characteristics mattering. A partial classification is:

- Random function approaches (Kushner, Mockus, Zilinskas) where the outcome of the function is considered a stochastic variable up to the moment it is evaluated. Actually it is a deterministic approach in the sense that no stochastic variables are used to generate new points, but a stochastic model is used to describe the (random) function given all former iterates. Perhaps this is most relevant for tackling noisy objective functions.
- Simple multi-start approaches, where multi-start is mixed with PRS (multi-singlestart) to generate random starting points.
- Clustering algorithms.
- Hit and run type of approaches where the next iterate depends on current iterate and an acceptance/rejection rule.
- Population based algorithms which generate offspring and select according to a fitness criterion.

6. Summary and Discussion Points

- Results of investigation of SGO algorithms consist of a description of the performance appropriate to the algorithms (parameter settings) and characteristics of test functions or function classes.
- The target of an assumed user and what is considered as success is needed to obtain good performance criteria.
- The performance graph is a useful instrument to compare algorithms.

[2]For this a probability model is needed or simple considerations such as can be found in the early Karnopp result, see [9]. Measuring and using the information in the algorithm usually leads to more extended algorithms with additional parameters complicating the analysis of what constitutes good parameter settings.

- Relevant characteristics depend on the type of algorithm and performance criterion under consideration.
- Landscape has to relate to both the algorithm and the function being optimized.
- Algorithms, to be comparable, must make use of the same information, accuracies, principles etc.
- It is wise to include simple benchmark algorithms like PRS, PAS and Multi-start as references.
- A generic algorithmic description for all SGO algorithms does not exist.

References

1. Baritompa, W.P., Mladineo, R.H., Wood, G.R., Zabinsky, Z.B. and Baoping Zhang (1995), Towards pure adaptive search. *Journal of Global Optimization* 7, 73–110.
2. Boender, C.G.E. and Romeijn, H.E. (1995), Stochastic methods. In: Horst, R. and Pardalos, P.M. (eds.), *Handbook of Global Optimization*, pp. 829–871, Kluwer, Dordrecht.
3. Dixon, L.C.W. and Szegö, G.P. (eds.) (1975), *Towards Global Optimisation*, North Holland.
4. Hendrix, E.M.T. and Roosma, J. (1996), Global optimization with a limited solution time. *Journal of Global Optimization* 8, 413–427.
5. Hendrix, E.M.T. (1998), Global optimization at work. Ph.D. thesis, Wageningen Agricultural University.
6. Hendrix, E.M.T. and Klepper, O. (2000), On uniform covering, adaptive random search and raspberries. *Journal of Global Optimization* 18, 143–163.
7. Hendrix, E.M.T., Ortigosa, P.M. and García, I. (2001), On success rates for controlled random search. *Journal of Global Optimization* 21, 239–263.
8. Horst, R. and Pardalos, P.M. (eds.) (1995), *Handbook of Global Optimization*, Kluwer, Dordrecht.
9. Karnopp, D.C. (1963), Random search techniques for optimization problems. *Automatica* 1, 111–121.
10. Klepper, O. and Hendrix, E.M.T. (1994), A method for robust calibration of ecological models under different types of uncertainty. *Ecological Modelling* 74, 161–182.
11. Patel, N.R., Smith, R. and Zabinsky, Z.B. (1988), Pure adaptive search in Monte Carlo optimization. *Mathematical programming* 43, 317–328.
12. Törn, A. and Zilinskas, A. (1989), *Global Optimization. Lecture Notes in Computer Science 350*. Springer-Verlag, Berlin.
13. Zabinsky, Z.B. and Smith, R.L. (1992), Pure adaptive search in global optimization. *Mathematical Programming* 53, 323–338.