Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

Two Strategies of Adaptive Cluster Covering with Descent and Their Comparison to Other Algorithms

D. P. SOLOMATINE
International Institute for Infrastructural, Hydraulic
and Environmental Engineering (IHE)
P.O.Box 3015, 2601 DA, Delft, The Netherlands
tel. +31 15 2151815, email: sol@ihe.nl

Abstract

Two strategies of randomized search, namely adaptive cluster covering (ACCO), and adaptive cluster covering with descent (ACD), are introduced and positioned in the group of the global optimization techniques. Several algorithms based on these new strategies are compared with other techniques of global randomized search in terms of effectiveness, efficiency and reliability. The other techniques include two versions of multistart, two versions of the controlled random search (CRS2 and CRS4) and the canonical genetic algorithm. Thirteen minimization problems including two parameter identification problems (for a flexible membrane mirror model and a hydrologic model) are solved. The algorithm ACCO, and a version of CRS4 algorithm (Ali and Storey 1994) show the highest efficiency, effectiveness and reliability. The second new algorithm, ACD, is in some runs very efficient and effective, but its reliability needs further improvement.

Keywords: global optimization, adaptation, clustering, covering, descent, parameter identification.

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

Two Strategies of Adaptive Cluster Covering with Descent and Their Comparison to Other Algorithms

D. P. SOLOMATINE

In the present paper two strategies of global search are considered - adaptive cluster covering (ACCO) and adaptive cluster covering with descent (ACD). In order to show its relation to the existing techniques, we give a brief overview of related techniques and compare the performance of the two algorithms to other existing methods.

Posing an optimization problem

A global minimization problem (GOP) with box constraints is considered: find an *optimizer* x^* such that

$$f^* = f(x^*) = \min_{x \in X} f(x) \qquad \textbf{(1)}$$

where the objective function *f(x)* is defined in the finite interval (box) region of the *n*-dimensional Euclidean space:

$$X = \{x \in \mathbb{R}^n : \boldsymbol{a} \le x \le \boldsymbol{b} \quad \text{(componentwise)}\ \} \qquad \textbf{(2)}$$

This *constrained optimization* problem can be transformed to an *unconstrained optimization* problem by introducing the *penalty* function with a high value outside the specified constraints. In cases when the exact value of an optimizer cannot be found, we speak about its *estimate* and, correspondingly, about its *minimum estimate*.

Approaches to solving the problem (1)-(2) depend on the properties of *f(x)*. The following cases may be considered:

(1) *f(x)* is a single- or multi-extremum analytically expressed function;

(2) *f(x)* is a single-extremum function which is not analytically expressed;

(3) no assumptions are made about the properties of *f(x)*, so in general case it is multi-extremum function which is not expressed analytically.

An important class of problems where multi-extremum optimization helps, includes problems of *parameter optimization (identification, calibration)* for a model of a physical system where the objective is to identify the values of model parameters that are not known *a priori*. This is achieved by feeding the model with input data, and comparing the computed output variables with the values measured in the physical system. The difference should be minimized by solving an optimization problem in which the independent variables are the unknown model parameters. Given the vector OBS_t of observed values for the output variables and the corresponding vector MOD_t of the modelled values at time moments $t = 1...T$, one of the widely used evaluation criteria for the *discrepancy (error)* between the model results and observations is:

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

$$E = \frac{1}{T} \sum_{t=1}^{T} w_t \, (OBS_t - MOD_t)^{\gamma} \tag{3}$$

where γ is taken between 1 and 2, and w_t is the weight associated with particular moments of time (for γ=2 and w_t=1, E is the mean square error). *Duan et al. 1993* and *Pintér 1995* report the high degree of non-linearity and often the non-smoothness of such functions.

In practice, the GOP of calibration is sometimes solved simply by applying single-extremum function minimization (*local search*). In other problems when an analytical expression of an objective function is available and derivatives can be computed then gradient-based methods are used (*Jacobs 1977*). In the case when no analytical expression is given and derivatives cannot be computed direct search methods can be used such as downhill simplex descent (DSD) (*Nelder and Mead 1965*) or direction set method (*Powell 1964*). *Brent 1973* and *Press et al. 1990* describe the modification of Powell's method, where the line minimization is achieved by bracketing and quadratic interpolation. It has been modified for the use in constrained optimization problems by the introduction of a penalty function outside the constraints interval. This latter method is referred to in this paper as the Powell-Brent method.

Many of the engineering applications done in the 1970s and 1980s used accepted methods for single-extremum function minimization, but often without the investigation of unimodality. Most researchers recognized the problem of the "good" initial starting point, and even mentioned the necessity of trying several such points, but the consideration of the rigorous practical procedures have been outside their attention. This can be partly attributed to the lack of awareness within the engineering community of developments in the area of global optimization.

Approaches to the multi-extremum minimization

The reader is referred to *Archetti and Schoen 1984, Rinnoy Kan and Timmer 1984, Törn and Žilinskas 1989, Zhigljavski 1991, Shoen 1991, Pintér 1995* for an extensive coverage of various methods. Many algorithms aimed at global search include various ideas so that no accurate grouping of methods is possible. However, for the purposes of this paper we may distinguish the following groups:

- set (space) covering techniques;
- random search methods;
- methods based on multiple local searches (multistart);
- evolutionary and genetic algorithms;
- other methods (simulated annealing, trajectory techniques, tunneling approach, analysis methods based on a stochastic model of the objective function).

We will briefly consider some other methods of randomized search which will be used for comparison with adaptive strategies ACCO and ACD introduced later.

Controlled random search (CRS)

Price (1978, 1983, 1987) proposed several versions of algorithm called *controlled random search* in which the vector r^k is generated on the basis of a randomly chosen subset of previously generated points. The basic idea is that at each iteration a simplex is formed from a sample and a new trial point is generated as a reflection about the centroid of the remaining points; the worst point in the original set is replaced then by the new trial point. The Nelder-Mead-type local search is incorporated in the version *CRS3* of *Price 1987* The ideas of Controlled random search algorithms have been further extended by *Ali and Storey 1994a* producing CRS4 and CRS5; see also subsequent publications mentioned in *Törn WWW 1997*.

The versions of the controlled random search implemented and tested in this study were *CRS2* (*Price 1983*) and a modified version of *CRS4* (*Ali and Storey 1994a*). In *CRS4* the non-uniform Hammersley distribution, which provides "more uniform" sampling, is used for the initial sample. When the new best point is found, it is "rewarded" by an additional search around it in which points from beta-distribution are sampled. *CRS4* was modified using for the initial sample uniform distributions rather than the Hammersley distribution. The modification of *CRS* is referred to here as *CRS4a*. It appeared that on several functions *CRS4a* showed better results than the original *CRS4*. This however, can be attributed to differences in implementation. *CRS2* and *CRS4a* are used in the comparisons below.

Multistart and clustering

The basic idea of the family of *multistart* methods is to apply a local search procedure several times, and then to make an assessment of global optimizer. The comparison of algorithms undertaken in this paper will include two multistart algorithms - *Multis* and *M-Simplex*. They are both constructed according to the following pattern:

Step 1. Generate a set of *N* random points and evaluate the function *f* at each of these points.

Step 2 (*reduction*). Reduce the initial set by choosing *p* best points (with lowest value of *f*).

Step 3. (*local search*). Launch local search procedures starting from each of the *p* points. The best point reached is the minimizer assessment.

In *Multis*, the Powell-Brent local search (see *Powell 1964, Brent 1973, Press et al., 1990*) is used at step 3. *M-Simplex* is similar to *Multis*, but instead of Powell-Brent search the downhill simplex descent of *Melder and Nead 1965* is used.

Popular versions of multistart and other GO algorithms are based on a priori clustering of potential starting points. These can be found in *Becker and Lago 1970,* and *Törn 1973* and *1978*. The *region (area) of attraction* of a local minimum x^* is the set of points in *X* starting from which a given local search procedure *P* converges to x^*. In the ideal case, the multistart methods aim at beginning this local search procedure exactly once in every region of attraction (*Törn and Žilinskas 1989*).

Two typical problems are connected with clustering methods. Firstly, clusters may contain several areas of attraction and the local search procedure may miss the global minimum; such a situation can be termed *underclustering*. Secondly, one region of attraction may be divided over several clusters. In this case the local search procedure will locate the same minimum more than

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

once with corresponding redundant function evaluations; this situation may be called *overclustering*. Both ACCO and ACD strategies use clustering as the first step of randomized search.

Evolutionary strategies and genetic algorithms

The family of *evolutionary algorithms (EA)* is based on the idea of modelling the search process of natural evolution, although these models are crude simplifications of biological reality. EAs introduce some modifications to random search and use terminology from biology and genetics. For example, for a random sample at each iteration pairs of parent individuals (points) selected on the basis of their 'fit' (function value) recombine and generate new 'offspring'. The best of these are selected for the next generation. Offspring may also 'mutate'. The idea is that fit parents are likely to produce even fitter children. In fact, any random search may be interpreted in terms of biological evolution: generating a random point is analogous to a mutation, and the step made towards the minimum after a successful trial may be treated as a selection.

Historically, evolutionary algorithms have been developed with three variations: evolution strategies (ES), evolutionary programming (EP), and genetic algorithms (GA). *Back and Schwefel 1993* give an overview of these approaches, which differ mainly in the types of mutation, recombination and selection operators. A canonical *genetic algorithm* (*Holland 1975*, *Goldberg 1989*, *Michalewicz 1996*) has received lately a lot of attention, and not only in professional publications. Applications of evolutionary algorithms, especially GAs, are found in many areas; see, for example, *Wang 1991, Cieniawski 1995, Gulsen et al 1995.*

Several versions of GAs were compared (including versions with the 'tournament' and 'fitness rank' selection) and the one that performed best was chosen for the present study. This is a variant of GA with the 'fitness rank' selection, one-point crossover, 15-bit coding of variables, bit mutation, and preservation of the best points discovered so far. A complex termination condition is employed, involving the fractional proximity of the found minimum estimate to the averaged function value of the predetermined percentage (20%) of the population, the number of iterations without improvement in the function value, and the total number of iterations. In each generation checks were made for the appearance of repetitive points to prevent redundant re-evaluations.

The Strategy of adaptive cluster covering

The basic ideas behind this approach were first outlined by *Solomatine 1995*. They are described here in more detail. The strategy of adaptive cluster covering (ACCO) was first conceived as a workable combination of some common sense ideas that could be useful in general-purpose search. Most of these ideas such as reduction, clustering and covering, have been discussed in the literature on global optimization for a long time, and used in different methods of search.

Becker and Lago (1970) used clustering in GO for the first time, shortly after the idea had been developed by A.Törn. Subsequently it was used in many algorithms (*Törn and Žilinskas 1989*). The algorithm of Becker and Lago (*BL*) used the subsequent clustering of points in subregions with reduction of samples. The idea of subsequent reduction is indeed similar to that

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

used in ACCO. However, the difference is that clustering is used only once in ACCO, and then instead of clustering, the subsequent covering is applied. The ACCO strategy is based on the following four principles.

1. Clustering. Clustering is used in multistart algorithms to identify regions of attraction and to launch procedures of single-extremum search in each region. However, the situation when clusters contain several areas of attraction and the local search procedure launched once will miss the global minimum (what is called here *underclustering*) may easily happen, and papers describing this class of multistart algorithms normally warn the possible user.

In the proposed strategy, clustering is also used for the purpose of identifying promising regions. But instead of assuming the existence of a local minimum and launching the local search, it is proposed that these regions should be used as subdomains of the objective function, and that global search is continued by covering in each subdomain.

2. Covering shrinking subdomains. The idea of covering is used in set covering algorithms, and in pure direct random search. In the considered strategy, the type of covering is not fixed as, say, grid (passive), active or purely random (uniform). In the version of an algorithm described below, random covering of each subdomain is used; that is, the values of the objective function are assessed at points drawn from the uniform or some other distribution. The procedure of covering is repeated many times for the subdomains that are progressively reduced in size.

3. Adaptation. Adaptive algorithms update their algorithmic behaviour depending on new information revealed about the problem under consideration. ‘Behaviour’ is influenced by a number of different things: parameters, like step or standard deviation; converging properties (stopping rules), or even various techniques and approaches used. In the case of the global search problems the algorithmic behaviour can change from step to step as, for example, in the case where the assessment of the function variation rate (Lipschitz constant) or the expected position of the global optimum is changing. In the considered strategy, the implementation of adaptation is in *shifting* the subregion of search, *shrinking* it, and changing the density (number of points) of each covering, depending on the previous assessments of the global minimizer.

4. Periodic randomization. Due to the probabilistic nature of points generation any strategy of randomized search may simply miss a promising region for search (similar to the situation when local search procedures may miss the global minimum). In order to reduce this danger it is reasonable to re-randomize the initial population, that is, to solve the problem several times or (and) to re-randomize some populations at intermediate steps.

Depending on the implementation of each of these principles it is possible to generate a family of various algorithms using non-rectangular domains (hulls), non-uniform sampling and various versions of cluster generation and stopping criteria. These may be suitable for certain situations such as different types of objective functions. In the present work one algorithmic implementation is presented (ACCO-1), so the abbreviation ACCO will be used both for the strategy and for the algorithm.

Algorithm ACCO (ACCO-1) - implementation of the Strategy of adaptive cluster covering

0. $l = 0$.

1 (*initial sampling*). Sample uniformly an initial population of N points in feasible domain X.

2 (*initial reduction*). Compute the function value f_i at each point and reduce the population by choosing p best points (with lowest f_i).

3 (*initial clustering*). Identify k_N clusters, such that the points inside a cluster are “close” to each other, and the clusters are “far” from each other. For each cluster, identify the smallest region (*n*-dimensional interval or a hull) for the subsequent search containing all points from the cluster. Set current region number $k = 1$. Set regional iteration number $e = 1$.

4 (*start of subsequent regional iteration e*). Sample r_k points inside region k, evaluate f at each of them, and choose s_k best points creating the set R_k. Reduce the region so that it includes the best points only.

5 (*shifting* to the center of attraction). Identify the ‘center of attraction’ of the region. This could be the best point or the centroid of the best subset. Shift the region so that its center coincides with the center of attraction.

6 (*shrinking*). Reduce the size of the region so that its linear size would be v_k% of the previous one.

7 (*stopping criteria for the current regional iteration*). Check criteria:

- C_1 is achieved when a fixed number e_1 of regional iterations is reached;
- C_2 is achieved if average function value for the best u% points in R_k does not differ (fractionally) more than w from the same value computed at the previous iteration $k-1$;
- C_3 is achieved if during the last e_2 regional iterations, there was no improvement of the minimum estimate.

```
If ($C_1$ or $C_2$) and $C_3$, then begin
    if $k = k_m$, then go to 8;
    prepare processing of the next cluster (region): set $k = k + 1$;  $e = 1$;
end
else begin
    prepare the subsequent regional iteration: Check criterion $C_{2B}$ (the average
    function value in the regional set $R_k$ is larger than the average of all points
    evaluated so far, that is, the region seems to be ‘non-interesting’)
    if $C_{2B}$ then
        (‘non-interesting’ region) considerably decrease the sample size $r_k$ for the
        next iteration by $rd_k$ % (e.g., 30%)
    else
        (‘interesting’ region) slightly decrease $r_k$ by $ri_k$ % (e.g., 5%);
    set $e = e + 1$.
end;
Go to 4.
```

8 (*final accurate search*). Construct the region with the linear size of q% of the domain interval around the best point found so far. Perform shifting and shrinking (steps 5 and 6) in a repetitive fashion until the stopping criterion C_1 or C_2 is satisfied.

9. Check wether $l= T$. If yes, then STOP, otherwise set $l=l+1$ and go back to step 1.

Figure 1 shows the initial sampling, and the regional iterations 1 and 2 for cluster k=1 in a two dimensional case. For functions with “curved” surfaces a non-rectangular shapes for subregions (e.g., hulls of best points subsets) may be more efficient. Other versions of the algorithm with this feature and which use sampling from the beta-distribution are currently being explored. At the moment, however, however, no assumptions were made about the Lipschitz constant, so the issue of accuracy is left open for future research. A potentially useful idea is to

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

draw conclusions periodically from the values of the current density and the assessments of the Lipschitz constant to determine when to stop.

Returning to the *BL* algorithm of *Becker and Lago 1970*, described also in *Törn and Žilinskas 1989* and which is ideologically close to ACCO, we must state that

- in BL, the global search in subregions is based on the subsequent multiple recursive clusterizations in each initial cluster. Indeed, in some cases the number of local optima is very high and such approach is then justified; however *a priori* the properties of the function surface are not known, and it is not necessary to find all local optima. For practical problems it is more reasonable to use the subsequent covering combined with reduction and shrinking subregions;
- in BL, as the number of points in subsamples is reduced, it may easily happen that the volume of the corresponding subregions will not decrease, or only decrease slowly due to the existence of 'average' points close to the boundaries. ACCO takes a more direct approach by forcing the volume to decrease stepwise; at the same time, reduction is applied to get rid of 'bad' points.

Choice of parameters. The ranges for parameter values used in the experiments were: initial population N = 30...1000; reduced initial population p = 40...300; number of clusters k_N = 3...20 depending on p; initial size r_k of a cluster sample variable starting from the initial number of points in the cluster; number s_k of points to which r_k is reduced equal to r_k; percentage of best points for criterion C_2: u = 30 % ; number of regional iterations e_1 = 3...15; minimum number of regional iterations made without improvement in function value: e_2 = 2...5; w = 1...5; v = 70...95 % ; T = 1...5. In the particular experiments reported below, the following values were used: e_2 = 2, T = 1, v=75%, u=50%, w=0.01, *rd*=5%, *ri*=30%, q = 15%. The maximum number of function evaluations was not limited.

A critical parameter appeared to be the number of regional iterations e_1. When it was too low it prevented the global minimum being reached, and when it was too large then number of function evaluations (f.e.) increased leading to a minimum that was normally reached anyway in subsequent steps. The following rule was found reasonable to use: for $2 \leq n \leq 4$, e_1 is set to 3, for higher n e_1 is set to n div 10 + 4. An important parameter is the number of generated points r_k at each iteration. Setting it to the original number of points in the cluster appeared to work well in most cases.

Other critical parameters were the size of initial population N and the number of clusterized points p. It was decided to keep the linear dependency between N and the dimension n; and between p and n. For comparisons the following rule was used: with n rising from 2 to 30, N increases from 50 to 300, and p from 40 to 200 respectively.

In the version of *ACCO* incorporated in the GLOBE system the user can set the parameters manually. In the next version it will be possible to choose one of the several sets of *parameter set rules (PSR)* which has been constructed on the basis of multiple experiments with *ACCO*.

In steps 7 and 8, the stopping rule C_2 is used to check if there is no improvement in the function value. The rule C_1 limits the number of iterations ensuring that the search will not take too long, and the rule C_3 (connected to C_1 and C_2 by AND operator) ensures that the search in a certain region is not dropped prematurely. Finally, the rule C_{2B} is made responsible for limiting the search efforts in 'non-interesting' regions. Different sets of PSRs may represent different strategies of search that a user would like to follow, depending on how expensive (long) is one evaluation of f. For example, the user may want to specify the approximate total number of function evaluations allowed. The rules will then identify the necessary values for the number of clusters k_N , the number of the sample size in the iteration r_k, and the number e_1 of regional

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

iterations. It must be stated that more experiments are needed to find a reasonable compromise between the mentioned parameters.

ACCOL strategy: the combination of ACCO with the multiple local searches

In this strategy of global search, referred to as ACCOL (adaptive cluster covering with local searches) there are two phases:

1. ACCO phase. ACCO strategy is used to find several regions of attraction represented by the promising points ('potent points') which are close to potential minimizers. The potent set P_1 is formed by taking the best point from R_k found for each cluster during the progress of ACCO. After ACCO stops, the set P_1 is reduced to P_2 by identifying several *m* (1...4) best points which are distant from each other, with the distance at each dimension being larger than, say, 10% of the range for this dimension;
2. *Local search (LS) phase*. An accurate algorithm of local search is started from each of the potent points of P_2 (multistart) to find accurately the minimum. The version of Powell-Brent search was used.

The advantage of such approach compared with traditional multistart is the significant economy in function evaluations. This is due to:

- the significant decrease in the dangers brought by *underclustering*;
- significantly less function evaluations needed by *ACCO* to process one cluster, than one direct LS;
- the elimination of the most 'non-interesting' potent points from P_1 and consequently the significant reduction in the number of the necessary multistarts made because *ACCO* moves the representatives of areas of attraction (potent points) much closer to potential minimizers -;
- the faster convergence of LS because *ACCO* moves the starting potent points much closer to potential minimizers.

Adaptive cluster covering combined with downhill simplex descents

The comparative experiments with several GO algorithms have shown the efficiency and effectiveness of the *ACCO* and *ACCOL* algorithms. These experiments have also shown certain attractive features of the downhill simplex descent (DSD) of *Nelder and Mead 1965*. This method appeared to perform well in a region which was not too close to the local optimizer, so that just several steps of DSD could be used in order to move 'good' points considerably closer to an optimizer.

The algorithm combining the positive features of both approaches has been built and called *adaptive cluster covering and descent* or, in short, *adaptive cluster descent (ACD)*. Its basic idea is also to identifying the area around the possible local optimizer using clustering, and then apply covering and DSD in this area. The main steps of ACD strategy are:

- sample the points (e.g., uniformly) and reduce the sample to contain only the best points;
- cluster the points, and reduce the clusters to contain only the best points;

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

- in each cluster, apply the limited number of steps of DSD to each point, thus moving them closer to an optimizer;
- if the cluster is potentially ‘good’, that is, contains points with low function values, cover the proximity of several best points by sampling more points, for example, from the uniform or beta distributions;
- apply local search such as DSD or some other algorithm of direct optimization starting from the best point in ‘good’ clusters. In order to limit the number of steps, the fractional tolerance is set to be, say, 10 times higher than the final tolerance, that is, the accuracy achieved is the average. Then apply the final accurate search, again DSD, starting from the very best point reached so far. The resulting point is the assessment of the global optimizer.

This strategy was realised in the form of the following algorithm.

Algorithm ACD (ACD-1) - implementation of the Strategy of adaptive cluster descent

1 (*initial sampling*). Sample uniformly the initial population of N points in feasible domain X.

2 (*initial reduction*). Compute the function value f_i at each point and reduce the population by choosing p best points (with lowest f_i).

3 (*initial clustering*). Identify k_N clusters, such that the points inside a cluster are close to each other and the clusters are far from each other. For each cluster identify the smallest region (n-dimensional interval or hull) for the subsequent search, which contains all points from the cluster. Set the current region number $k = 1$.

4 (*reduction*). Check the criterion C_{2B} which states that the average function value in the regional set R_k is lower than the average of all points clustered in step 3 (‘interesting’ region). If C_{2B} then

reduce the sample R_k by *rd1* %; set the number of descent steps to *ds1*;

else (‘non-interesting’ region)

reduce the sample R_k by *rd2* %; set the number of descent steps to *ds2*;

5 (*several steps of downhill simplex descent*). For each point in R_k perform *ds* steps of DSD. In R_k replace all points by the new points reached as a result of these descents.

6 (*region update*). Construct the new region containing all points from R_k .

7. (*additional covering*) If C_{2B} (‘interesting’ region) then select cp_k points and sample cr_k points in the ϵ-proximity of each of them. This ϵ is taken as percentage of the size of the region containing R_k. Evaluate f in all of them and add these points to R_k .

8 (*intermediate downhill simplex descent*). For the best point in R_k perform DSD until the predefined intermediate accuracy is reached. Store the newly reached point in the set *R2*.

9. Increase k by 1. If $k \le k_N$, goto 4.

10. (*final downhill simplex descent*). For the best points from *R2* perform DSD until the predefined final accuracy is reached. The resulting best point is the global optimizer assessment.

The author is grateful to a reviewer who pointed out the similarity of the ideas behind this algorithm to those of an early multistart algorithm proposed by *Törn 1978*. In his algorithm the parallel local searches are stopped repeatedly, the working points are clustered and a reduced number of processes from each cluster are resumed. This difference is similar to that between ACCO and BL: instead of repetitive clustering the method advocated in this paper uses covering instead.

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

Choice of parameters. In *ACD* there are less parameters to choose than in *ACCO*. The parameters *N* and *p* depend linearly on dimension *n*, like in *ACCO*. The critical parameters appeared to be the numbers of DSD steps *ds1, ds2.* If they are too high, the number of f.e. increases considerably. However, experiments showed that the proximity of a 'good' point reached in this way is usually reached anyway at a later stage at steps 7 - 10. Another critical parameter is the fractional tolerance of the intermediate DSD in step 8. If it is small then there is an increase in f.e., again without much gain.

The values for parameters were chosen following multiple experiments with the algorithm in terms of delivering the acceptable effectiveness and efficiency. All parameters were investigated but mainly with the parameters *rd1, rd2, ds1* and *ds2* used in criterion C_{2B}, and the following values were chosen: *rd1* = 50%, *ds1* =4, *rd2* = 67%, *ds1* =2. Parameters cp_k and cr_k were taken as being equal to 3 and 5 respectively. The ϵ-proximity for covering was set to be 33% of the region containing the remaining points in the cluster. For the intermediate DSDs the fractional tolearance of 0.1 was used, with the fractional tolerance of 0.01 used for the final DSD.

ACDL algorithm: combination of ACD with the multiple local searches

Following the idea used in ACCOL as described above it is now possible to construct the strategy of global search, referred to as ACDL (adaptive cluster covering and descent with local searches). This consists of two phases:

1. *ACD phase*. The ACD strategy is used to find several regions of attraction represented by the promising points that are close to potential minimizers ('potent' points). The potent set P_1 is formed by taking one best point from R_k found for each cluster during the progress of ACD. After ACD stops the set P_1 is reduced to P_2 by selecting only several *m* (1...4) best points which are distant from each other, with the distance at each dimension being larger than, say, 10% of the range for this dimension;
2. *Local search (LS) phase*. An accurate algorithm of local search is started from each of the potent points of P_2 (multistart) to find accurately the minimum. As in *Multis* and *ACCOL*, the version of the Powell-Brent non-derivative minimization is used. The value of LS in ACDL is less than in ACCOL, since the final step in ACD is the accurate local search DSD. These multiple Powell-Brent searches normally allow the achievement of a slightly higher accuracy than with DSD.

Changing the algorithm in the process of search (*structural adaptation*) used in ACCOL and ACDL has shown the high efficiency and effectiveness of such a scheme. This is because they combine the efficiency of the first stage (ACCO or ACD) in identifying quickly the minimum assessment and the effectiveness of local search which, if started from the point close to the minimum, reaches the minimum with the high degree of accuracy.

Global optimization tool *GLOBE*

The PC-based system GLOBE has been built by the author to apply the global optimization techniques to the problems of model calibration. GLOBE can be configured to use an external program as a supplier of the objective function values. The number of independent variables and the constraints imposed on their values are supplied by the user in the form of a simple text file. Figure 2 shows the organization of the calibration process. *Model* must be an executable module (program) which does not require any user input. The user has to supply two transfer programs *P1* and *P2*. These three programs (*Model*, *P1*, *P2*) are activated from GLOBE in a loop. GLOBE runs in DOS protected mode (DPMI) providing enough memory to load the program modules. It can be downloaded from *www.ihe.nl/hi*.

The user interface includes several graphical windows displaying the progress of minimization in different coordinate planes projections. The parameters of the algorithms can be changed easily by the user.

Currently GLOBE includes several algorithms, of which the following algorithms of randomized search are included for comparison:

S *CRS2 (controlled random search,* by *Price 1983)* described above;
S *CRS4a* (our modification of the *controlled random search* by *Ali and Storey 1994a)*;
S *genetic algorithm (GA)*;
S *Multis* - multistart algorithm, as described above;
S *M-Simplex* - multistart algorithm, as described above;
S *adaptive cluster covering* (*ACCO*);
S *adaptive cluster covering with local search* (*ACCOL*).
S *adaptive cluster descent* (*ACD*);
S *adaptive cluster descent with local search* (*ACDL*).

It is planned to complement GLOBE by such methods as simulated annealing, topographical multilevel single linkage (*Törn and Viitanen 1994, Ali and Storey 1994b*) and other reportedly efficient methods.

Comparing nine algorithms

Comparison of algorithms proposed by different authors is always a difficult task since so much depends on the details of implementation. Quite naturally, authors try to design the test suite in a way that would show the best features of the proposed algorithm. In spite of that, testing is always useful since it gives general idea about the algorithms' performance and may point to classes of functions where this or that algorithm excels. Every effort was made to make the comparison as fair as possible, despite the possible differences in various parameters and stopping conditions.

The nine algorithms were tested on a number of runs using a set of several optimization problems. These included the traditional benchmarks functions used in optimization with known global optima (*Rosenbrock 1960, Dixon and Szegö 1978, , Griewank 1981, Duan et al. 1993*), and the two problems of model calibration where the global optima were unknown (Table 1).

The first calibrated model was a model of a *flexible membrane mirror* (*Vdovine et al. 1995*) with 3 parameters. The model is based on a finite-difference solution of simultaneous partial differential equations describing the deflection of a round membrane under the influence of a magnetic field created by three concentric actuators. The problem is to identify the voltages for these actuators that fit the shape of the membrane to a predefined parabolic shape. The second calibrated model was *the hydrologic rainfall-runoff model SIRT* (a so-called, conceptual lumped model with 8 parameters, a version of the *Sugawara 1978* model). The model was run on data for a Latin American river basin. Calibration pushed the model accuracy to its limit: lower error values can be obtained only with a more complex model.

The number *N* of points in the initial sample and the number of points in the reduced sample were chosen according to the rule that these numbers must grow linearly with the dimension *n*, from *N*=50 at *n*=2, to *N*=300 at *n*=30. For *CRS2* and *CRS4a* the formula recommended by their authors is *N=10(n+1)*. This gives *N=30* at *n*=2, and *N*=310 at *n*=30. In *ACCOL*, *ACDL*, *Multis* and *M-Simplex* the fractional tolerance of 0.001 was used. The reason of using this fractional tolerance lies in the fact that in the problem of model calibration quite high function values may be returned, and the use an absolute tolerance normally leads to a very large number of f.e. with minimal gain in function value. Experiments with CRS were based on absolute tolerances in order to have a comparison base to experiments reported in the literature.

One of the important objectives of this work was the comparison of GA to algorithms based on other ideas explored in GO. Since GA uses discretized variables (we used the 15-bit coding, i.e. the range is 0...32767) an accurate comparison would only be possible if the values of the variables for other algorithms were discretized in the same range as well. This has been done for *ACCO, ACD, CRS4*. Other algorithms, including the local search stages of *ACCOL* and *ACDL*, use real-valued variables. Our intention was to investigate the general behaviour of the algorithms, important from the practical point of view, rather than studying their accuracy very close to the global minimum. In order to keep the variance of resulting performance indicators as low as possible the method of mutually-dependent tests, widely accepted in simulation experiments, was used; all algorithms are fed from the same initial random population of points.

Performance indicators

Three main performance indicators were investigated:

- *effectiveness* (how close the algorithm gets to the global minimum, or, if the latter is unknown, how low its found estimate is);
- *efficiency* (running time) of an algorithm includes not only the total time spent on function evaluations but also the running time of the algorithm itself. However, the latter is negligible compared with the former, especially in parameter identification problems when one function evaluation requires running what is often a complex model. This is why efficiency is measured by the number of function evaluations needed;
- *reliability* (robustness) of algorithms can be measured by the number of successes in finding the global minimum, or at least approaching it sufficiently closely. An algorithm may be not too efficient, but if at every single run it finds practically sufficient estimate of a global minimum, it is considered a reliable one.

Effectiveness and efficiency

The plots on Figure 3 show the progress of minimization for some of the problems from Table 1 (other results can be seen on *www.ihe.nl/hi/sol/p_jogo/allplots.htm*). The results are averaged across 5 runs. This was produced as follows. After each run a plot of the function values against the number of f.e. was produced. Then for the consecutive f.e. divisible by 50 the corresponding values for function values were averaged across all runs and plotted. Such averaging was done while all the runs were in progress, that is, until at least one of the runs stopped because of a stopping condition (even though other runs were continuing), and *the last but one* point was selected to represent the situation. However, *the last point* was added to the

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

plot to represent the best function value found through all five runs. Some of the line segments between the last two points are vertical. This means that the best function value has been reached in one of the runs *earlier* than shown by the abscissa of the last but one point. Note that most points of *ACCOL* plot correspond both to *ACCO* and *ACCOL,* and only some of the last points correspond to the local search phase of *ACCOL*; the same applies to *ACDL* and *ACD.*

The comparison results can be summarized briefly as follows. For the functions of 2 variables, *ACCOL*, *CRS4a* and *M-Simplex* are the most efficient, that is, faster in getting to the minimum. In the flexible mirror model, Hosaki, Rastrigin and Six-hump camelback functions *M-Simplex* quite unexpectedly showed the best results. With the Rosenbrock function *ACDL* was also among the best methods, however, not with other functions.

With functions of higher dimensions, which are more more interesting for the purposes of this paper, *ACCOL* and *CRS4a* again performed best, and had similar performance. *M-Simplex* was the worst with all Shekel 4-variable functions, but was even a bit better than *ACCOL* and *CRS4a* with Hartman 3-variable and 6-variable functions. *ACDL* was on average as the third best in performance after *ACCOL* and *CRS4a*. It is a ‘slow starter’. However, on some runs *ACDL* showed very high efficiency. *GA* is the least efficient method, and is ineffective with all Shekel functions.

Multis and *CRS2* are both effective, reaching the global minimum in most cases, but much less efficient than other algorithms.

It is worth mentioning that the total number of f.e. made by algorithms and the balance between effectiveness and efficiency depends significantly on the several user-defined parameters, namely, number of clusters, cluster size, number of iterations, etc. In GLOBE, these can be set by the user.

Reliability (robustness)

Reliability can be measured as the number of successes in finding the global minimum with the predefined accuracy. Because of the randomized character of search no algorithm can be 100% reliable. For most functions of 2 variables most algorithms were quite reliable (with the exception of GA, which was often converging prematurely). Only the Rastrigin function with many equidistant local minima with almost equal values presented difficulties.

With the functions with more than two variables the situation was different. It can be seen from Fugure 3 that for most algorithms the ordinate of the last point can be considerably less than the ordinate of the previous point. This means that the least function value was found in just one or two of the runs, but not in all of them. The *CRS2* and *Multis* algorithms appeared to be the most reliable for functions of higher dimensions but was by far the least efficient. *ACDL* was, however, not always reliable even though it showed efficiency on some runs.

As seen from Figure 3, in all cases, except for *GA* the minimizer estimate is normally quite close to the global minimum. Small differences could be attributed partly to the way the real-valued variables were coded. A more accurate statistical analysis of single-start failure probabilities has yet to be done.

Discussion

It is seen from the results of comparisons that the *CRS2* algorithm which is permanently oriented towards the whole function domain has to perform more function evaluations, that is, it has low efficiency. *Multis* showed similar properties.

The lower efficiency of *GA* can be attributed to the type of ‘crossover’ used (exchange of some of the parents’ coordinate values) which often leads to redundant evaluations of the ‘offspring’ in the search space quite far from their highly fit parents, and hence normally with the lower fitness. So the fitness gained by the parents is not inherited by many of their offspring. It was also found that *GA* often converges prematurely, especially in the variant with tournament selection. Wether this feature is inherent to the whole class of evolutionary algorithms following the ideas of natural evolution, which are indeed quite appealing but highly redundant, or wether it is just the feature of the version of a GA implemented in this study, has yet to be investigated. It is worth mentioning that reportedly the efficiency of GAs can be increased by using other types of crossover, like intermediate recombination in evolutionary strategies. Examples given in *Back and Schwefel 1993* show the higher efficiency of evolution strategies which reportedly better preserve from generation to generation the identified structure of the objective function.

The relatively higher efficiency of *ACCOL* and *CRS4a* can be explained by their orientation towards smaller search domains which is especially efficient for high dimensions. For example, the reduction of the linear size of a box in 30-dimensional space by half reduces its volume by more than a billion times. *ACDL* on some runs has shown high efficiency but its reliability was not the best.

It is interesting to mention that due to the relative freshness of the approach and creative ‘marketing’ genetic algorithms are currently receiving a lot of attention and are applied to many multi-extremum optimization problems. In practically all areas where GAs are currently used most of the other global optimization algorithms could be used as well.

Conclusions

1. Two strategies and corresponding algorithms of randomized search, namely, adaptive cluster covering (*ACCOL*) and adaptive cluster descent (*ACDL*), were introduced and compared to several other algorithms for several functions.

2. *ACCOL* was shown to have high effectiveness, efficiency and reliability. On average, the performance of *ACCOL* was comparable to that of *CRS4a* and *CRS4* (*Ali and Storey 1994a*), reportedly a very efficient algorithm. In many practical problems one function evaluation is expensive (slow), and their total number is then the critical parameter. The experience shows that in this case *ACCO* (without the local search phase) would be the first choice to obtain a reasonable optimizer assessment. In most cases *ACCO* was faster than other algorithms in getting quite close to the value of the global minimum. However, more research is needed in fine-tuning the *ACCOL* parameters.

3. On the basis of experience with many practical problems where the functions of high dimensions and expensive evaluations are involved other tested algorithms - *CRS2*, multistart of Powell-Brent (*Multis*), and genetic algorithm (*GA*) cannot be generally recommended for complex problems with ‘expensive’ functions. These algorithms either do not normally reach a global minimum, as in the case of *GA*, or require many more function evaluations. If, however, a high number of f.e. is not a critical parameter, *Multis* in many instances can reach an accurate solution. Multistart of the classical downhill simplex descent (DSD) (*Nelder and Mead 1965*) with a preliminary reduction (*M-Simplex*) performs very well with the functions of low variables. However with the functions of higher dimensions it often converges prematurely to a local minimum.

4. The attempt to merge the attractive properties of downhill simplex algorithm with *ACCO* in the *ACDL* algorithm proved to be efficient and effective on some of the runs with functions of higher dimensions. However, more accurate tuning of its parameters is needed to improve its reliability.

5. The experiments confirmed the usefulness of

(1) combining an efficient (fast) random search algorithm with the subsequent effective (accurate) local search. In the present study, the *ACCO* and *ACD* algorithms were combined with the modified Powell-Brent method of direct non-derivative optimization and included in the *ACCOL* and *ACDL* algorithms;

(2) DSD, in combination with the reduction and multistart (implemented in *Multis*), performs very well for small dimensions and low number of local optima. In combination with *ACCO* (implemented in *ACDL*) it shows potential for the problems of higher dimensions.

6. The choice between various methods of global optimization may depend on the type of problem, and more research is needed to compare reportedly efficient methods like simulated annealing, evolution strategies, topological multilevel linkage among others (see *Ali and Storey 1994b; Törn and Viitanen 1994; Locatelli and Schoen 1996*). The best results can probably be achieved by *structural adaptation*, that is, switching in the process of search between different algorithms.

Acknowledgments

The author is grateful to M.B. Abbott, and R.K. Price for the fruitful discussions, and to the author's affiliation institute IHE for the support of this research direction. Special thanks go to the three anonymous reviewers for the very valuable comments and constructive suggestions.

References

Ali, M.M, and Storey, C. Modified controlled random search algorithms. Intern. J. Computer Math., 53, pp. 229-235, 1994.

Ali, M.M, and Storey, C. Topographical multilevel single linkage. J. of Global Optimization, 5, pp. 349-358, 1994.

Archetti, F. and Schoen, F. A Survey on the global optimization problem: general theory and computational approaches, in Annals of Operations Research 1, J.C.Baltzer A.G. Publishing., pp 87-110, 1984

Back, T., and Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation, 1, No. 1, pp. 1-23.

Brent, R.P. Algorithms for Minimization without Derivatives. Prentice-Hall, Englewood-Cliffs, N.J., 195p., 1973

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

Cieniawski, S.E, Eheart, J.W. and Ranjithan, S. Using genetic algorithms to solve a multiobjective groundwater monitoring problem. Water Resour. Res., 31 (2), 399-409, 1995.

Dixon, L.C.W. and Szego, G.P. (eds.). Towards global optimization, North-Holland, Amsterdam, 472p., 1978.

Duan, Q., Gupta, V., Sorooshian, S. Shuffled complex evolution approach for effective and efficient global minimization. J. of Optimization Theory and Applications, 76 (3), pp. 501-521, 1993.

Goldberg, D.E. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley, 1989.

Gulsen, M, Smith, A.E., Tate, D.M. A genetic algorithm approach to curve fitting. Int. J. Prod. Res, 33 (7), pp. 1911-1923, 1995.

Griewank, A.O. Generalized descent for global optimization. J. Optimiz. Theory Appl., 34 (1), 11-39, 1981.

Jacobs, D.A.H. The state of the art in numerical analysis, Academic Press, London, 1977.

Locatelli, M. and Schoen, F. Simple linkage: analysis of a threshold-accepting global optimization method. J. of Global Optimization, 5, pp. 95-111, 1996.

Michalewicz, Z. Genetic algorithms + data structures = evolution programs. Springer, Berlin, 1992.

Nelder, J.A, and Mead, R. A simplex method for function minimization. Computer Journal, vol. 7, No. 4 p. 308-313, 1965.

Neumaier, A. WWW page *http://solon.cma.univie.ac.at/~neum/glopt.html*, World Wide Web, 1997.

Pintér, J. Global optimization in action. Kluwer, Amsterdam, 1995.

Powell, M.J.D. An efficient method of finding the minimum of a function of several variables without calculating derivatives. Computer Journal, 7, 155-162, 1964.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. Numerical recipes in Pascal. The art of scientific computing. Cambridge University Press, Cambridge, 759p., 1990.

Price W.L. Global optimization by controlled random search. J. Optimiz. Theory Appl., 40, 333-348, 1983.

Rinnoy Kan A.H.G. and Timmer G.T. Stochastic global optimization methods. I. Clustering methods. II. Multilevel methods. Mathematical programming, 39 (1), p.27-56 and 57-78, 1987.

Rosenbrock, H.H. An automatic method for finding the greatest or least value of a function. Computer J., 3, 175-184, 1960.

Schoen, F. Stochastic techniques for global optimization. J. of Global Optimization, 1, pp. 207-228, 1991.

Solomatine, D.P. The use of global random search methods for models calibration, Proc. XXVIth Congress of the International Association for Hydraulic Research (IAHR), vol.1, pp. 224-229, London, September 1995.

Sugawara, M. Automatic calibration of the tank model, Proc. Intern. Symposium on Logistics and Benefits of Using Mathematical Models of Hydrologic and Water Resource Systems, Italy, October 24-26, IIASA, Laxenburg, Austria, 1978.

Törn, A. A search clustering approach to global optimization, in Towards global optimization, 2, pp. 49-62. North-Holland, 1978.

Törn, A., and Žilinskas, A. Global optimization. Springer-Verlag, Berlin, 1989, 255pp.

Törn, A. and Viitanen, S. Topographical Global Optimization using pre-sampled points. J. of Global Optimization, 5, pp. 267-276, 1994.

Törn, A. WWW page *www.abo.fi/~atorn/globopt.html*, World Wide Web, 1996.

Vdovine, G., Middelhoek, S., Bartek, M., Sarro, P.M., and Solomatine, D.P. Technology, characterization and applications of adaptive mirrors fabricated with IC-compatible micromachining. Proc. SPIE's Intern. Symposium on Optical Science, Engineering and Instrumentation, Conference 'Adaptive Optical Systems and Applications', vol. 2534/13, San-Diego, USA, July 10-14, 1995.

Wang, Q.J. The genetic algorithm and its application to calibrating conceptual rainfall-runoff models. Water Resour. Res., 27 (9), 2467-2471, 1991.

Zhigljavsky, A. Theory of global random search. Kluwer Academic Publishers, Dordrecht, 341p., 1991.

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

Figures captions

Fig. 1. ACCO in 2-variables case. Domain is randomly covered, points are evaluated, ‘good’ ones are clusterized; for each cluster, smaller regions are repetitively formed around the currently ‘best’ point and progressively covered: (a) initial sampling and clustering; (b) regional iteration 1 with the ‘best’ point indicated; (c) regional iteration 2 with the new ‘best’ point indicated.

Fig. 2. Organization of the calibration process using GLOBE.

Fig.3. Averaged performance of algorithms in five runs.

Published in: Journal of Global Optimization, 1999, vol. 14, No. 1, pp. 55-78.

Table 1. Functions used in comparing algorithms.

Ref on Figure 3	Function	Dimension	Number of optima	Value of the global minimum
	Rosenbrock	2	1	0.0
	Hosaki	2	2	≈ -2.338
a	Rastrigin, shifted by 2.0	2	>50	0.0
	Six-hump camelback (Branin), shifted by 1.036285	2	6	0.0
b	Goldstein-Price function	2	4	3.0
c	Flexible mirror model error	3	?	≈ 0.0
	Hartman3, shifted by 3.32	3	4	≈- 0.6
d	Hartman6, shifted by 3.32	6	4	≈ 0.0
	Shekel5, shifted by 10.5364	4	5	≈ 0.0
	Shekel7, shifted by 10.5364	4	7	≈ 0.0
e	Shekel10, shifted by 10.5364	4	10	≈ 0.0
f	Griewank function	10	>1000	0.0
g	SIRT Rainfall-runoff model error	8	?	< 47.0