RAIRO. RECHERCHE OPÉRATIONNELLE

PIERRE COURRIEU

A distributed search algorithm for global optimization on numerical spaces

RAIRO. Recherche opérationnelle, tome 27, n° 3 (1993), p. 281-292

<http://www.numdam.org/item?id=RO_1993__27_3_281_0>

Recherche opérationnelle/Operations Research
(vol. 27, n° 3, 1993, p. 281 à 292)

A DISTRIBUTED SEARCH ALGORITHM FOR GLOBAL OPTIMIZATION ON NUMERICAL SPACES (*)

by Pierre COURRIEU (1)

Communicated by Jacques CARLIER

Abstract. – *This article presents a new algorithm that searches for the global extrema of numerical functions of numerical variables. This "Distributed Search" algorithm builds an evolving "visiting" probability distribution on the search domain, and the process converges towards states in which the probability density is maximal on the neighborhood of the target extrema. The convergence of the algorithm is demonstrated. Then its performance is tested on some "hard" test functions and compared to that of a recent, well-known algorithm.*

Keywords: Global optimization, distributed random search.

Résumé. – *Nous présentons dans cet article un nouvel algorithme pour la recherche des extrema globaux de fonctions numériques de variables numériques. Cet algorithme, dit de « Recherche Distribuée », fait évoluer une distribution de probabilités de visite sur le domaine de recherche suivant un processus convergeant vers des états où la densité de probabilité est maximale au voisinage des extrema recherchés. La convergence est démontrée, et les performances sont testées sur des problèmes difficiles, en comparaison avec un algorithme récent et réputé.*

Mots clés : Optimization globale, recherche aléatoire distribuée.

1. INTRODUCTION

An optimization problem usually involves a numerical function defined on a domain of a given search space (the space of variables). The idea is to find the point or points in the search domain where the function takes on an extreme value (global minimum or maximum, depending on the case). A direct algebraic solution is rarely available, and a simple approach consisting of systematically scanning the search domain with a selected step is generally not usable, since the size of the search domain (in precision units) increases exponentially as the number of variables increases.

(*) Received November 1991.

(1) C.R.E.P.C.O. (U.R.A. C.N.R.S. n° 182), Université de Provence, 29, avenue Robert-Schuman, 13621 Aix-en-Provence, Cedex 1, France.

Recherche opérationnelle/Operations Research, 0399-0559/93/03/$ 4.00

Optimization problems vary in difficulty, depending on the properties of the function to be optimized. Uniextremal functions are generally the easiest to optimize because "local search" procedures can be used. Such procedures (relaxation methods, gradients and conjugate gradients [6]) are efficient, and guarantee finding the solution. But the functions encountered in practice usually have multiple extrema, making it necessary to use "global search" methods, which can generally only guarantee finding the solution with a given degree of probability. Extensive efforts have been made within the past few years to solve hard optimization problems. Certain algorithms rely on the principle of a converging random walk based on Markov chains and Gibbs distributions (Metropolis algorithm [12]. Simulated Annealing [5, 10, 15]). In other methods global distributions of probabilities represented by samples of points belonging to the search domain are made to converge (Genetic Algorithms, [7, 9]). In still other methods, convergence is achieved by the global distribution control of a set of local searches (Multistart Algorithms, Clustering and Multilevel algorithms [1, 13, 14]). Other approaches exist, although most of them appear to be more interesting from a theoretical point of view than a practical one (*see* [16]).

Presented below is a new global optimization algorithm based on a "distributed search" principle. A previous version of this algorithm was presented in [2] and applied with success in [3] and [11]. The present version is more elaborate.

2. DISTRIBUTED SEARCH ALGORITHM

The algorithm presented here is applicable to the search for the global extrema of a numerical function f defined on a bounded domain χ of $\mathbb{R}^n$ or of a discrete subset of $\mathbb{R}^n$. It is designed to take advantage of certain global features of functions at different successive scales of analysis (these features becoming "local" at finer scales). The basic idea is very simple: Let $X1$ and $X2$ be two points in χ such that $f(X1)<f(X2)$. Provided the function is not too irregular, then there exists a set of points X in a neighborhood of varying size around $X1$ such that $f(X)<f(X2)$. At the limit, if the neighborhood is reduced to $X1$ itself, then this is necessarily true. The algorithm's task is to sample χ according to a convergent distribution of probabilities controlled by estimation of appropriate neighborhood scales at the various stages of the process.

The function need not have any special properties for the basic algorithm to be applicable, and it suffices to be able to evaluate the function at all

points in the search domain. The presentation given below includes a variant of the algorithm, which can be selected via the Boolean variable DLS (Directional Local Search). This variant theoretically requires the function to be C^1 continuous, but in practice this is not necessary since the gradients used can be replaced by simple approximations for non-smooth functions and empirical functions. Indeed, it is quite sufficient to approximate the partial derivatives with the corresponding finite increase ratios since, by definition:

$$\lim_{h \to 0} \frac{f(x_1, \cdots, x_k + h, \cdots, x_n) - f(x_1, \cdots, x_k, \cdots, x_n)}{h} = \frac{\partial f}{\partial x_k}.$$

The algorithm is presented in Pseudo-Pascal. The following conventions are used: intervals are shown in the usual notation, real interval bounds are separated by two periods (..). The term "random" followed by an interval denotes a random value taken from a uniform distribution of probabilities on the specified interval. The algorithm shown here concerns the search for minima, but it would of course apply to a search for maxima if the opposite of the function were used (*i. e.* $f := -f$).

```
{ Parameters: DLS: Boolean selector of the secondary regulation type; M: sample size; α>0:
speed parameter; ε>0: arbitrarily small constant; }
{ Sample initialization: use an M×n cell matrix and an M f-value vector }
opt :=1;
for j : =1 to M do
  begin
  for i: =1 to n do x_i(j) := random [min x_i, max x_i];
  evaluate f (X(j));
  if f (X(j))<f (X(opt)) then opt := j;
  end;
{ Scale initialization (in order to correctly cover the search domain) }
for i: =1 to n do s_i := (max x_i − min x_i) / (2 M^(1/n) tg (π (0.5)^(1/n)/2));
{ Progression}
b =0; T :=M/10;
repeat
w :=0; k :=0;
for i :=1 to n do d_i : =0;
  repeat
  k :=k+1;
  p :=random [1..M]; q :=random [1..M];
  if f (X(p))>f (Xq)) then swap q and p values;
  if random [0, 1[<b then begin { variant }
                                X := X (p) − rΔf (X (p));
    { Determine the step length r using a standard bisection
      method (see [16], pp. 21-22). This procedure returns f (X) }
          end { variant }
        else begin
          for i :=1 to n do
            x_i :=s_i tg (π random]−1/2, 1/2[)+x_i(p);
    { if necessary replace X by its projection in the search domain }
          evaluate f (X);
```

```
            end;
  if f(X)<f(Xq)) then begin { winning trial }
                                  X(q) :=X; f(X(q)) := f(X).
        w :=w+1;
        for i :=1 to n do d_i :=d_i+(x_i(p)-x_i)^2;
        end;
  if f(X(q))<f(X(opt)) then opt : =q;
  until (w=T) or (k=M);
if DLS then begin { variant } b :=(T-w)/2 T; c :=1; end { variant }
        else c :=w/T;
if w>0 then for i :=1 to n do s_i :=(c/πα) (d_i/w)^(1/2)+ε;
until (stopping rule);
```

2.1. The parameters

The Boolean parameter DLS selects a secondary regulation approach which occasionally enters into the process. The regulations are explained below. The parameter M is the size of the sample of points used by the algorithm to encode a distribution of visiting probabilities on the search domain χ. The greater the sample size M, the more complex the encodable distributions, so the complexity of the function to be optimized must be considered when the value of this parameter is chosen (a minimum of 100 or so is recommended for multiextremal functions). Note that function complexity should not be assessed solely on the basis of the number of local extrema. We shall see in the experimental section below that there are functions with an infinite number of local minima which are easy to optimize with a distributed search, provided that good global properties are available. The parameter α controls the overall progression speed, and also affects the activation probability of the secondary regulations (usual range: $0.5 \leq \alpha \leq 1$). The parameter ϵ has little effect in practice, but it guarantees that the s_i scales will not drop to zero and thereby prevents the process from freezing indefinitely. ϵ is generally chosen to be smaller than the desired precision for the result.

2.2. Generation of points

The coordinates of the visited points are ordinarily generated independently by means of the generating function: $x_i = s_i \operatorname{tg}(\pi u_i) + m_i$, with u_i taken at random from a uniform distribution on $]-1/2, 1/2[$. The random variable defined as such obeys an n-dimensional Cauchy law with density:

$$g_n(X;\ m,\ S) = \prod_{i=1}^{n} \frac{1}{\pi s_i} \frac{1}{1 + ((x_i - m_i)/s_i)^2}.$$

The point m is the center (mode and median) of the distribution, and the scale parameters, the s_i's, are the quartile deviations. Cauchy's law has no

moments, and in particular its variance is infinite, which is a reflection of the fact that the density decreases slowly and is never negligible. Cauchy variables have properties which seem particularly favorable for the method. In particular, we tested exponentially decreasing densities (e. g. logistic or Gaussian) and found that the algorithm was difficult to tune and that its performance was not as good. However, other point-generating methods can be effectively used when the problem exhibits certain constraints. In particular, the generating density can have a bounded support, provided it never goes to zero on the search domain. And it does not necessarily have to be isotropic.

A second type of generating process can occasionally be used when the DLS variant is selected. In this case, a new point X is generated simply by a directional local search step from a point $X(p)$ selected from the sample. Note that in this case, it is perfectly useless if not undesirable to do a complete local search, since it would most likely, and laboriously, lead to a local extremum. Moreover, this does not correspond to the regulation function the process was supposed to fulfill.

2.3. Regulations

The primary regulation is done by the estimation of neighborhood scales (square roots of the mean $d_i's$), which outputs "winning trials" at each step of the process. This estimation makes it possible to adjust the scale parameters (the $s_i's$) of the generating functions while also taking the chosen progression speed into account (determined by the parameter α). The smaller the α, the larger the generation scales relative to the progression scales, such that the number of winning trials tends to decrease and the exploration of the domain is less dependent on the current sample, and thus less risky. Choosing a small α also increases the probability that the secondary regulations, which are useful in case of difficulty, will be activated.

It can just so happen that at a certain analysis scale, the function to be optimized exhibits some "bad" properties such as numerous poorly differentiated basins. In this case, the scale values are generally overestimated by the primary regulation. The process tends to stagnate for some time, and then start up again, but time has been uselessly wasted and the resulting scale estimations are based on winning samples which are too small (the normal sample size is set at $T=M/10$). The secondary regulations take effect when the winning try rate goes below 10%. These regulations force the process to probe basins whose scale is smaller than the current analysis scale, efficiently solving the stagnation problem. In the basic version of the algorithm, this is

simply done by multiplying the generation scales by the ratio (c) between the observed number of winning trials (w) and the expected number of wins (T). In the DLS variant, the secondary regulation is obtained by increasing the probability (b) of the directional local search steps as a function of the failure rate (the upper limit of b is set at 0.5). This variant can be used to process the function's directional local properties when they are relevant (which, by the way, is not the general case for multiextremal functions).

2.4. Stopping rules

Various stopping rules can be used, depending on the requirements for the application in question. One can simply limit the number of function calls (consumption criterion), or wait until the generation scales are close to ϵ (precision criterion on χ). One can also wait until the dispersion of the function values in the sample is close to zero. The advantage of this last stopping criterion is that it avoids unwanted triggering of secondary regulations in cases where the process is coming to an end and the sample is still concentrated on one or more extrema with the same value.

2.5. Convergence

Although of minor interest from a practical standpoint, it might be useful for theoretical clarity to study the limits of convergence of the process. A simple and quick demonstration based on some general concepts found in [16] is given below.

Assume the search domain χ has a finite measure $\mu(\chi)$, and that χ is either discrete or the function is continuous at at least one optimum. Note that the first two conditions are always satisfied in practice since numbers are represented in finite form in the computer.

Assume that the visiting probability density at all points of χ has a non-zero lower limit of λ as the calculation time tends towards infinity. Let f_t be the value of the function at the point visited by the process at time t, and let χ_δ be the subset of points in χ on which the function differs from the optimal value by some quantity less than δ, where $\delta>0$. Under the general conditions stated above, this subset has a non-zero measure. The stochastic convergence can be expressed as:

$$\text{Prob}\{\exists t \leq k; | f_t - \min_{X\in\chi} f(X) | < \delta\} \geq 1 - \left(1 - \lambda \frac{\mu(\chi_\delta)}{\mu(\chi)}\right)^k \underset{k\to\infty}{=} 1.$$

(Replace min with max for a maximum search)

It suffices to show for the Distributed Search that the limit λ is not equal to zero. For the basic version of the algorithm, the visiting probability density associated with any point X in χ at time t is:

$$V_t(X) = \sum_{j=1}^{M} \frac{2}{M} P_t(f(X_j))\, g_n(X; X_j, S_t).$$

P_t is the cumulative probability function of the values of the function in the sample at time t if we are maximizing, and is its one's complement if we are minimizing. Multiplying by $2/M$, we obtain the probability that the j-th point in the sample will be taken as the center of the generation, the sum of the probabilities obviously being equal to 1. Since the search domain is bounded, it follows directly from the above expression that the density V has a non-zero limit λ on χ, provided none of the components of the scale vector S of the generation density g_n has a zero limit. But precisely this limit is set at $\epsilon > 0$.

In the DLS variant of the algorithm, the density V_t is expressed in a similar manner by replacing the density g_n by a combination of the following type: $(1-b_t)\, g_n(X; X_j, S_t) + b_t$ prob $\{ X \in \mathrm{DLS}(X_j) \}$, where $b_t \leq 0.5$ and $\mathrm{DLS}(X_j)$ is a set of nearly zero measure. Given that the density g_n always has a non-negligible weight, the conclusions are the same for the variant and the basic version.

CONCLUSION: The Distributed Search is stochastically convergent.

3. EXPERIMENTAL STUDY OF PERFORMANCE

3.1. Test functions

Presented below is an experimental study of the performance of the distributed search algorithm on some hard minimization problems. Most test functions used in the current literature usually only have a small number of local extrema, which severely limits the scope of the tests. We were nevertheless able to find two families of standard hard functions for the test: Csendes functions [4] (*see* also [16], p. 16) and Griewank functions [8] (*see* also [14], p. 76). Since these functions have extreme, highly contrasted properties, a third family of functions with more balanced properties was selected. This family will be called here the "wave" functions. The three families of functions were used with 2 and 10 variables.

Csendes test functions:

$$Cn(X) = \sum_{i=1}^{n} x_i^6 \left(2 + \sin \frac{1}{x_i}\right), \qquad -1 \leq x_i \leq 1.$$

Contrary to what might appear, this function is defined at X=0, precisely where it has its unique global minimum (0). The function possesses a countable infinity of local minima on the search domain, and the oscillation frequency tends towards infinity on the neighborhood of the global solution. This property makes it practically impossible to minimize by applying local searches. However, Csendes functions have very favorable global characteristics: they oscillate between two 6-degree convex hulls which approach each other on the neighborhood of the solution.

Wave functions:

$$W\,n,\ k\,(X) = \frac{1}{n} \sum_{i=1}^{n} 1 - \cos(k\,x_i)\,\exp(-x_i^2/2), \qquad -\pi \leq x_i \leq \pi.$$

Wave functions have their unique global minimum (0) at X=0. The number of local minima on the search domain is k^n (for k odd) or $(k+1)^n$ (for k even). Only k=10 was used in this study, which gave 121 local minima for n=2, and 2.5937×10^{10} local minima for n=10. The function oscillates between two exponential hulls of constant mean (1) whose distance from each other is maximal on the neighborhood of the solution. This is not a favorable global characteristic, although the lower hull is potentially a good source of information, provided the search process manages to move far enough into the basins.

Griewank test functions:

$$G\,n\,(X) \;=\; 1 + \sum_{i=1}^{n} x_i^2/d_n \;-\; \prod_{i=1}^{n} \cos\left(x_i/\sqrt{i}\right),$$

For n=2: d_n=200, $-100 \leq x_i \leq 100$. For n=10: d_n=4,000, $-600 \leq x_i \leq 600$. These functions have their unique global minimum (0) at X=0, and have a large number of local minima. They have many "trap" basins on a large area around the solution, where the global properties of the function seem to be quite unfavorable. However, in the neighborhood of certain points, Griewank functions have some special directional local properties. To get an idea of these properties, assign one of the variables a value like $x_i = (2\,k + 1)\,\pi\,\sqrt{i}/2,\ k \in Z$. In this case, the partial derivatives for all other variables point to the solution, which is a stable attractor for the variables that reach its neighborhood.

3.2. Reference algorithm

Because the experimental results for problems like these are rare in the literature, a reference algorithm was used; We chose the Dekkers and Aarts

algorithm [5], which is a recent version of the Simulated Annealing algorithm adapted to numerical spaces. It is like a sequential Multistart algorithm in that it uses multiple directional local search steps, whose global distribution is governed by a simulated annealing rule. The algorithm was implemented in compliance with the indications provided by the authors except that the stopping rule was simplified. This rule was replaced by a "temperature" thresholding, involving a complete local search starting from the best point found whenever the temperature fell below 10^{-8}. For the test functions used, this order of magnitude for the temperature was indeed found to correspond to a "stalling" point in the process, after which no progress was observed. We used the tuning values proposed by the authors (namely: m_0=100, χ_0=0.9, L_0=10, t=0.75), except for the speed parameter that usually had to be reduced since a δ of 0.1 proved to be unsuitable to problems this difficult. To determine δ, a trial and error procedure was used to obtain 10 successive runs with no error in the result for each problem. This search succeeded for three out of six problems (W2, G2, G10), but it was impossible, even once, to obtain an exact result for the remaining problems. The value used for δ, then, was simply the minimum value found in those problems which were solved (0.005).

3.3. Tuning the distributed search

The distributed search parameters DLS, M, and α were also initialized by trial and error until we obtained 10 successive runs with no error in the result of any of the problems. This criterion obviously did not imply that the probability of error was equal to zero, which *a priori* is impossible in finite time. The results were considered exact (for all algorithms) when the process found a function minimum equal to 0, within the precision range of the compiler (Turbo-Pascal 5.0 on a Compaq 486/33L computer). The parameter ϵ was set at 10^{-20}

3.4. Results

The results are presented in Table. For each algorithm and each problem, the table gives the values of the parameters and the mean number of function calls for the 10 runs (f calls). Each gradient calculation was counted as a calculation of the function. Also given are the mean error on the function value (f error) and the mean euclidean distance between the best point found and the global minimizer (X error). Standard deviations (with 9 degrees of

freedom) are shown in parentheses to the right of the corresponding means.

TABLE

Parameters, and mean performance on 10 successive runs.

Problem	Algorithm	Dekker & Aarts (1991)	Distributed Search
C2	parameters	δ=0.005	DLS false, M=100, α=1.00
	f calls	46,549 (13,149)	7,028 (949)
	f error	3.38E−15 (1.01E−14)	No error
	X error	2.42E−03 (1.65E−03)	
C10	parameters	δ=0.005	DLS false, M=200, α=1.00
	f calls	430,301 (41,406)	89,453 (2,304)
	f error	1.30E−10 (1.18E−10)	No error
	X error	3.76E-02 (4.76E−03)	
W2	parameters	δ=0.005	DLS false, M=100, α=0.75
	f calls	29,485 (11,534)	4,161 (371)
		No error	No error
W10	parameters	δ=0.005	DLS false, M=250, α=0.75
	f calls	221,752 (29,660)	119,799 (2,617)
	f error	0.210 (0.028)	No error
	X error	2.927 (0.494)	
G2	parameters	δ=0.005	DLS true, M=150, α=0.80
	f calls	52,793 (4,741)	5,712 (393)
		No error	No error
G10	parameters	δ=0.01	DLS true, M=300, α=0.60
	f calls	251,870 (10,208)	205,584 (4,084)
		No error	No error

The results are quite clear and highly predictable, given the properties of the various functions. The functions which exhibit the best global properties are the easiest ones for the distributed search, regardless of their local properties. The secondary regulations were never activated for functions C2, C10, W2, and G2, such that the choice of true or false for the DLS option had absolutely no effect on the results. For function W10, the secondary regulation was activated quite early in the process but the chosen option only had a minor effect, the only difference being that convergence was obtained an average of 10% faster with the "DLS true option". The most difficult function for the distributed search was G10 whose optimization clearly required the DLS true option for the secondary regulation. With DLS false, the process frequently became trapped by local minima where the function is approximately equal to 0.01. Note that the minimization of

Griewank functions is relatively easy for the Dekkers and Aarts algorithm; this is logical given its strong directional local component. Note finally that Rinnooy Kan and Timmer [14] reported some disappointing results concerning the minimization of the same Griewank functions by a "Multi Level Single Linkage" algorithm. The exact reasons for their failure are not very clear, however.

4. CONCLUSION

The above global optimization algorithm based on a distributed search principle appears to be well-suited to handle the global characteristics of functions at different scales. Moreover, the algorithm's DLS variant also efficiently handles the directional local characteristics of functions. The high performance levels obtained clearly surpass current standards in this field. Further research efforts are needed to develop an efficient way of determining the best parameter values for each problem and the constraints specific to each application. One potential approach would be to estimate and analyze the frequency spectrum of the function, but to date, the only available parameter estimation method is trial and error. Furthermore, attempts to generalize the distributed search method to various topological spaces such as the ones encountered in combinatorial optimization have not given rise to truly convincing results.

ACKNOWLEDGEMENTS

We would like to thank M. J. Fischer (Yale University), J. Véronis (CNRS, Marseille), and the anonymous reviewers for their constructive remarks concerning this work. Financial support was provided by DCAN (Marine Nationale), Ministère de la Recherche et de la Technologie, and Conseil Régional PACA.

REFERENCES

1. C. G. E. Boender, A. H. G. Rinnooy Kan, L. Stougie and G. T. Timmer, A stochastic method for global optimization, *Mathematical Programming, 22,* 1982, pp. 125-140.
2. P. Courrieu, *A Distributed Search Algorithm for Hard Optimization.* Technical Report TA9101, CREPCO, Université de Provence, 13621 Aix-en-Provence Cedex 1, 1991.
3. P. Courrieu, *A convergent generator of neural networks,* Technical Report TA9102, CREPCO, Université de Provence, 13621 Aix-en-Provence Cedex 1, 1991.
4. T. Csendes, *A simple but hard-to-solve global optimization test problem.* IIASA Workshop on Global Optimization, Sopron (Hungary), 1985.
5. A. Dekkers and E. Aarts, Global optimization and simulated annealing, *Mathematical Programming, 50,* 1991, pp. 367-393.

6. R. Fletcher and C. M. Reeves, Function minimization by conjugate gradients, *Computer J., 7,* 1964, pp. 149-154.
7. D. E. Goldberg, *Genetic Algorithms in Search, Opitimization, and Machine Learning,* Addison-Wesley, Reading, Massachusetts, 1989.
8. A. O. Griewank, Generalized descent for global optimization, *J. of Optimization Techniques and Application, 34,* 1981, pp. 11-39.
9. J. H. Holland, *Adaptation in Natural and Artificial Systems,* The University of Michigan Press, Ann Arbor, 1975.
10. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, optimization by simulated annealing, *Science, 220,* n° 4598, 1983, pp. 671-680.
11. R. Lengellé and T. Denoeux, *Optimizing Multilayer networks layer per layer without backpropagation,* paper presented at ICANN'92, Brighton (U.K.), September 4-7, 1992.
12. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth and A. H. Teller, Equations of state calculations by fast computing machines, *J. Chem. Phys., 21,* 1953, pp. 1087-1091.
13. A. H. G. Rinnooy Kan and G. T. Timmer, Stochastic global optimization methods. Part I: clustering methods, *Mathematical Programming, 39,* 1987, pp. 27-56.
14. A. H. G. Rinnooy Kan and G. T. Timmer, Stochastic global optimization methods. Part II: multi level methods, *Mathematical Programming, 39,* 1987, pp. 57-78.
15. D. Vanderbilt and S. G. Louie, A Monte Carlo Simulated Annealing approach to optimization over continuous variables, *J. of Computational Physics, 56,* 1984, pp. 259-271.
16. A. A. Zhigljavsky, *Theory of Global Random Search,* Kluwer Academic Publishers, Dordrecht, 1991.