Journal of Advances and Scholarly Researches in Allied Education
Vol. 16, Issue No. 1, January-2019, ISSN 2230-7540

Random Search Global Optimization Technique on Personal Computers

Neeta Sharma*

SNRL Jairam Girls College, Lohar Majra, Kurukshetra

Abstract – In this paper we present a Turbo Pascal code of the random search technique based on quadratic interpolation approach for determining the global optimal solutions of nonlinear optimization problems. The Pascal code based on the random search algorithm is specially designed for solving nonlinear optimization problems on Personal Computers. The algorithm is tested on a set of standard test examples taken from literature. The results are compared with the earlier results of the Fortran version designed for solving such problems on main frames such as DEC 2050 system.

----------X----------

1. INTRODUCTION:

In a nonlinear optimization problem: Min f(X), f : R^n → R, subject to X ∈ S C R^n, X^* is said to be its global optimal solution if f (X^*) ≤ f(X) for all X ∈ S. However, if f (X) ≤ f(X^*) for all X ∈ S ⋂ $N_\in$ (X^*), where $N_\in$(X^*) is a ∈ neighbourhood of X^* then X^* is called a local optimal solution. Determining the global optimal solution of an optimization problem is ordinarily more difficult as compared to determining its local optimal solution, since for X to be a global optimal solution the requirement f(X^*) ≤ f(X) has to be ensured in the entire feasible domain S, whereas for X^* to be a local optimal solution this requirement has to be ensured only in a small feasible neighbourhood of X^*. However, practical considerations sometimes demand that global solutions rather than local solutions be determined. A number of computational techniques are now being reported in literature which search for the global optimal solution of an optimization problem (see for instance Dixon (1) Boggs (2). In continuation with the series of our papers on algorithms for solving nonlinear global optimization problems Mohan and Shanker (3, 4, 5, 6, 7) reported the use of a Fortran code to solve a number of test examples on Dec. 2050 system of the Roorkee University Regional Computer Center. In (8) the algorithm was used to solve the problem of reliability optimization of complex systems. In (9, 10) it was used to determine the underground reservoirs of subsurface structures by inverting the observed gravity anomalies. In (11) it was used to determine the hypocentral parameters (spatial co-ordinates and time of occurance) of an earthquake. However, a need was felt to develop a computer code for running on Personal Computers. Most of the Fortran compliers do not provide an efficient routine for the generation of random numbers. Fortunately, Turbo Pascal provides an efficient random number generator. This paper presents a Turbo Pascal code for the use on Personal Computers. The Code has been tested on a number of test examples taken from literature and the results are reported here.

In section 2 the computational steps of the algorithm have been presented. In section 3 we present the Turbo Pascal code of the algorithm. Finally in section 4 we present our experience of using the Pascal code for the solution of some test examples and draw conclusions based on the present numerical study.

2. THE COMPUTATIONAL ALGORITHM:

In this section we present the details of the random search algorithm for global optimization. It uses only function evaluations and avoids all other mathematical operations such as evaluation of derivatives, which sometimes may not be easy and may not even be justified. It assumes no specific mathematical properties for the functions appearing n the problem. The computational algorithm tries to obtain the global optimal solution of a nonlinear constrained optimization problems.

The random search approach originally advocated by Price (12, 13) works iteratively in two phases. In the first phase, the global phase, the objective function, is evaluated at a number of randomly sampled feasible points. In the second phase the local phase, these points are manipulated by local searches to yield a possible candidate for global optima. Although, theoretically there is no guarantee that the algorithm will locate the exact global optimal

solution in each and every case (perhaps none of the existing algorithms can guarantee it), yet it does determine global optimal solution in most of the cases when used on test problems where global solutions are known. In cases where the algorithm fails to locate the global solution, it generally determines a solution which is quite close to the global optimal solution.

The algorithm generates randomly N points (say N = 10 (n + 1), where n is the number of unknown independent variables of the problem) and evaluates the objective function f(X) at these points. These points and their function values are then stored in an N by (n + 1) array A in such a manner that the point with the minimum function value is stored as L and the point with maximum function value is stored as M. In the local search three distinct points $B_1 = L$, B_2 and B_3 are randomly sampled from array A and a new trial point P is obtained by quadratic interpolation (i.e. P is the point of minima of the quadratic curve passing through B_1, B_2 and B_3). If P is feasible then f(P) is evaluated and compared with the current f(M). If f(P) < f(M) then the current M is replaced by P in array A and new L and M determined. If however, P is not feasible or f (P) ≥ f(M) then the trial is discarded and a new choice of B_2 and B_3 is made. The process is continued iteratively till all the points suitably cluster around the global optimal solution which is checked by testing $|(f(M) - f(L))/f(M)| < \in$ where $\in$ is some pre-assigned accuracy. During each local search a counter is used to keep record of the unsuccessful trials (i.e. cases when P is not feasible or f(P) ≥ f(M). If this counter exceeds a certain prescribed number the best n points of the current array A are retained and the rest of the N – n points are replaced by new randomly generated feasible points. The computational steps of the algorithm are:

Global Phase :

(i) Choose N, say N = 10 (n+1), random feasible points and evaluate the objective function at these points. Store these points and their function values in an N by (n+1) array A. Set ITER = 1, MULT = 1.

(ii) Out of these N points find M and L, the points with greatest and lowest function values f(M) and f(L), respectively. If $|f(M) - f(L))/f(M)| < \in$ stop with the message that L is the global minima, otherwise set IFAIL = 1 and goto (iii).

Local Phase :

(i) From the array A chose randomly three points B_1 = L, B_2 and B_3 and determine the next trial point using the relation

$$P = 0.5 \left(\frac{(R_2^2 - R_3^2) f(R_1) + (R_3^2 - R_1^2) f(R_2) + (R_1^2 - R_2^2) f(R_3)}{(R_2 - R_3) f(R_1) + (R_3 - R_1) f(R_2) + (R_1 - R_2) f(R_3)} \right)$$

Check if P satisfies the specified feasible range $a_i < x_i < b_i$ i = 1, 2, ... n or not. If yes go to (iv), otherwise set $p_i = b_i$ if $p_i \geq b_i$ *and* $p_i = a_i$ if $p_i \leq a_i$, i = 1, 2, ... n and go to (iv).

(ii) Check if P is feasible. If yes go to (v), otherwise go to (vi).

(iii) Find f(P). If f(P) > f (M) go to (vi), otherwise replace the current M by P in array A and set ITER = ITER + 1. If ITER > ITL (where ITL is some preassigned positive integer) go to (viii), otherwise go to (ii).

(iv) Set IFAIL = IFAIL + 1. If IFAIL > LAST (where LAST is some prassigned positive integer) go to (vii), otherwise go to (ii).

(v) If MULT > MLAST (where MLAST is some prassigned positive integer) go to (viii), otherwise set MULT = MULT + 1 and replace the worst N-n points (points with highest function values) of array A by new randomly generated feasible points, and go to step (ii).

(vi) Stop with the indication that the current L is the best point with the best function value which could be found so far by the algorithm.

The flow chart of the algorithm is presented in Fig. 1.

3. TURBO PASCAL CODE OF THE ALGORITHM:

In this section we present the Turbo Pascal code of the random search algorithm. On personal computers most of the Fortran compilers do not provide efficient random number generators. The function 'random' of Turbo Pascal has been used to generate random numbers between 0 and 1 which are then scaled to the upper and lower bounds of the variables. The seed for the generation of random numbers is also selected randomly by initially calling the function 'randomize' so the problem of selecting an unbiased seed for the generation of random numbers in Fortran code has been overcome in Turbo Pascal. Similarly, for choosing B_1 and B_2 from the current array A the function random (NBIG)' is used (Refer step (iii) above. Version 5.0 of Turbo Pascal has been used to develop the code for the algorithm which may be updated for higher versions accordingly. The code may be obtained from the authors on request.

4. SOLUTION OF TEST EXAMPLES AND CONCLUSIONS:

In this section we present our experience of using the Turbo Pascal code for solving 35 test examples appearing in literature. These problems are given in (7) wherein they have been used to evaluate the algorithm in Fortran on Dec. 2050 system of the Roorkee University Regional Computer Center. Each problem has been run ten times on PC-AT. The accuracy used in each case (except when stated otherwise) is $\in$ = 0.001. A trial is considered a success if the objective function value is within one percent of the known global optimal value. The percentage of successful runs, minimum, average and maximum number of function evaluations required to solve each of them is reported in Table 1. For comparison purpose the percentage of success for each problem reported earlier is also given in this table. The maximum, minimum and average function evaluations of successful runs for each problem are also depicted through a graph (Fig. 2).

It is observed that the greatest number of function evaluations required are 37126 in problem no. 33. On the other hand the minimum function evaluations required is 60 in problem no. 16 Problem no. 3 gave 30% success, problem no. 32 gave 50% success, problem no. 20 gave 70% success, and problem no. 24, 25, 26 gave 90% success whereas all the other problems could be solved with 100% success. The time taken by the Turbo Pascal code is also comparable.

Thus it may be concluded that since the algorithm is applicable to a variety of problems it can be used to solve nonlinear optimization problems on a Personal Computer with the help of Turbo Pascal code. However, our experience shows that theoretically speaking the algorithm is applicable to all problems but for practical problems with a large number of conflicting constraints it is unduly large. It works satisfactorily well even if the number of variables is large.

The author is thankful to Dr. Kusum IIT, Roorkee for her valuable guidance.

Table 1: Percentage of success and maximum number of function evaluations required to solve test examples

Problem Number	Number of function evaluations			1% of success	
	Maximum	Minimum	Average	Present algorithm	Earlier algorithm
1	150	97	124	100	100
2	304	140	208	100	100
3	277	115	180	100	100
4	832	172	344	100	100
5	240	122	153	100	100
6	181	127	153	100	100
7	2281	400	1102	100	100
8	871	692	742	100	100
9	924	585	716	100	100
10a	1520	183	518	100	100
10b	2039	950	1726	100	80
11a	996	367	678	100	90
11b	1098	298	462	100	100
12	1002	517	710	100	40
13	7889	2154	4355	30	40
14	215	145	175	100	100
15	4409	3202	3823	100	100
16	150	60	98	100	100
17	295	223	262	100	100
18	404	214	298	100	100
19	1573	475	930	100	100
20	4351	2668	3412	70	30
21	35961	30673	32403	100	100
22	415	285	338	100	100
23	4768	987	1773	100	100
24	523	217	312	90	100
25	9061	2121	4407	70	100
26	30890	12535	21642	90	30
27	17765	10348	10952	20	100
28	156	78	139	100	100
29	366	116	233	100	100
30	851	201	428	100	100
31	3310	1211	2714	100	100
32	13111	10276	11410	50	70
33	37126	22106	28220	100	100

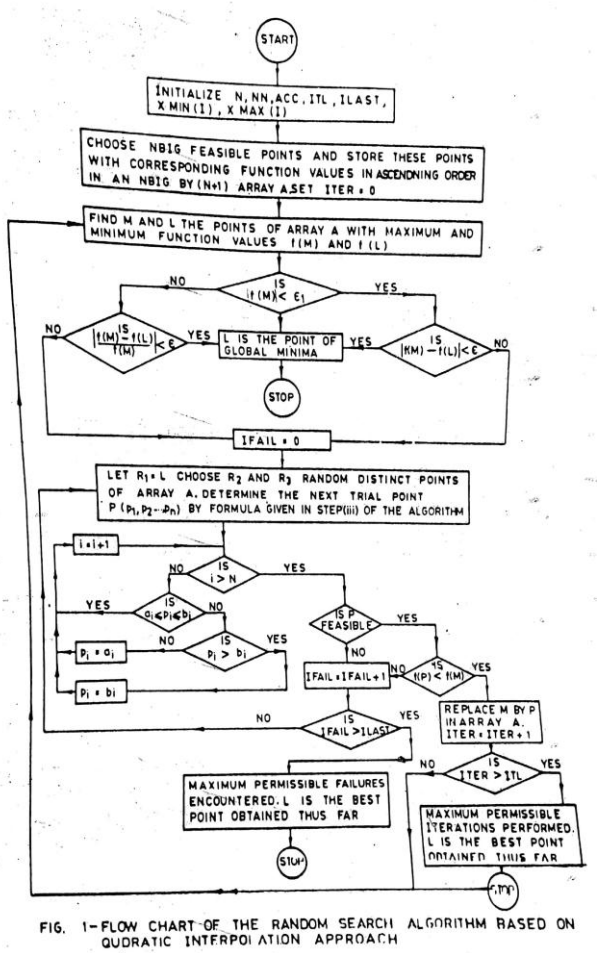
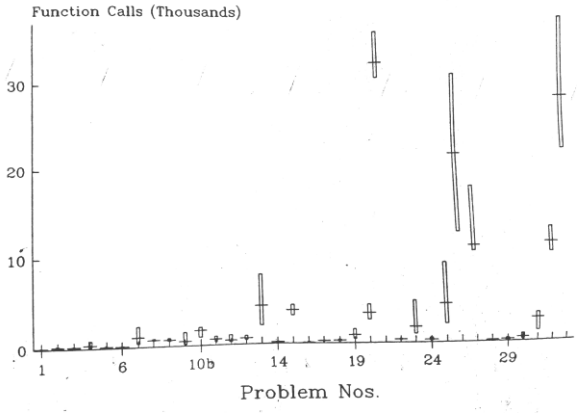


FIG. 1–FLOW CHART OF THE RANDOM SEARCH ALGORITHM BASED ON QUDRATIC INTERPOLATION APPROACH

Fig. 2.– Maximum, Minimum and Average Function Calls using Pascal Code on PC



208

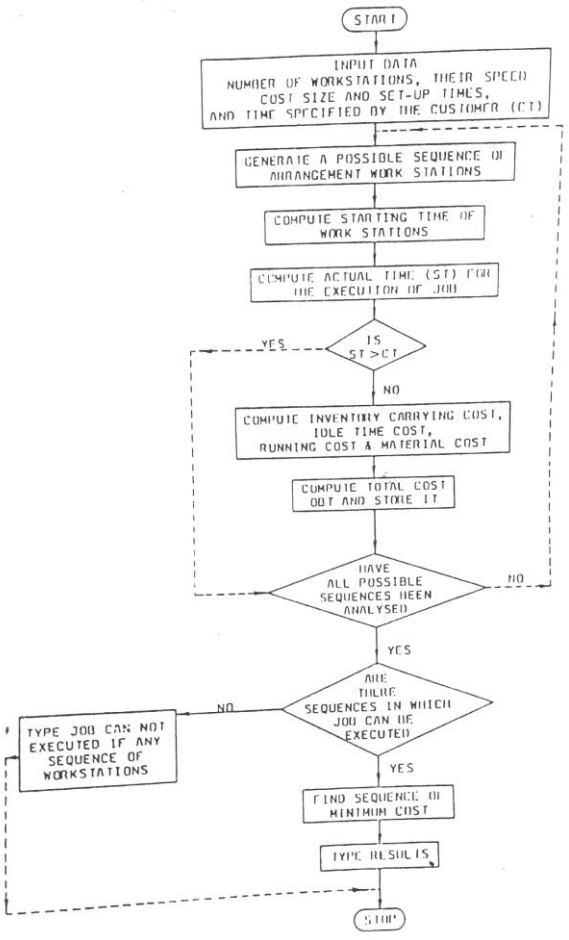


FIGURE 2

REFERENCES

1. Dixon, L.C.W., SZEGO, G.P. (1978). “Towards Global optimization-2”, North Holland, Amsterdam.

2. BOGGS, P.T., BYRD, R.H., SCHNABEL, R.B. (1985). “Numerical optimization SIAM, Philadelphia.

3. Mohan, C. K. SHANKER: "A numerical study of some modifier versions of controlled random search method for global optimization", International Journal of Computer Mathematics, U.K., Vol. 24, No. 1.

4. SHANKER, K., C. MOHAN (1986): "A computational algorithm for the global minima of constrained nonlinear optimization problems", Proceedings, Symposium on Optimization, Design of Experiments & Graph Theory, I.I.T. Bombay, India.

5. MOHAN, C., K. SHANKER (1990). "Computational algorithms based on random search technique for solving global optimization problems", International Journal of Computer Mathematics, U.K., Vol.33, pp. 115-126.

6. SHANKER, K., MOHAN, C. (1987). "A Random Search Technique for the Global Minima of Constrained Nonlinear Optimization Problems", Proceedings, International Conference on Optimization Problems and Their Applications, Singapore, p. 905.

7. SHANKER, K., C. MOHAN: "A Random Search Technique for Global Optimization Based on Quadratic Interpolation Approach, communicated to Journal of Optimization Theory & Applications.

8. MOHAN, C., SHANKER, K. (1988). "Reliability Optimization of Complex Systems using Random Search Techniques", Micro electrons and Reliability, vol. 28, No. 4, pp. 513-518.

9. SHANKER, K., C. MOHAN, K.N. KHATTRI (1987). "A random search technique of nonlinear optimization for the inversion of gravity data", Presented at the IV Indian Geological Congress, held at University of Roorkee, in February.

10. SHANKER, K., C. MOHAN, K.N. KHATTRI: "Inversion of gravity data by random search technique", communicated to the Journal of Association of Exploration Geophysics.

11. SHANKER, K., C. MOHAN, K.N. KHATTRI: "Inversion of seismological data using random search global optimization technique", communicated to Tectono physics.

12. PRICE, W.L. (1978). "A Controlled Random Search procedure for global optimization", in Towards Global Optimization 2, (eds.)

L.C.W. Dixon and G.P. Szego, North Holland, Amsterdam.

13. Price, W.L. (1983). "Global optimization by controlled Random Search", Journal of Optimization, Theory & Applications, Vol. 40, No. 3, pp. 333-348.

Corresponding Author

Neeta Sharma*

SNRL Jairam Girls College, Lohar Majra, Kurukshetra