International Journal of Computational and Applied Mathematics
ISSN 1819-4966 Volume 4 Number 1 (2009), pp. 1–9

https://dx.doi.org/10.37622/IJCAM/4.1.2009.1-9

A Comparative Performance of Swarm Intelligence Optimization Method and Evolutionary Optimization Method on Some Noisy Numerical Benchmark Test Problems

Sanjeev Kumar Singh[1] and Munindra Borah[2]

[1]Department of Mathematics & Computer Sc.
Union Christian College,Ri-Bhoi-793122, Meghalaya, India.
09436100837(M) 0364-2570152(Fax)

[2]Department of Mathematical Sc.,
Tezpur University, Tezpur, Assam, India.

Abstract

Recently many algorithms have been developed which mimics the natural procedure better known as Evolutionary Algorithm and claims to perform better than others. The objective of this paper is to test the performance of Genetic algorithm and Repulsive Particle Swarm method on some benchmark functions. Since GA(Genetic Algorithm) mimics the nature and (RPS) exploits the swarm intelligence, it will be interesting to see the performance of these two methods on the certain test functions. A brief idea of these functions are given in this section are as follows. These functions are also represented by graph to facilitate conceptualization of the nature of these functions by visual means.

Keywords: Genetic Algorithm, Particle Swarm Optimization, Genetic Programming (GP), Ant Colony System(ACO),Simulated Annealing, Tabu Search, Memetic Algorithm, External Optimization(EO).

Introduction

Optimization is central to any problem involving decision making, whether in Mathematics, Engineering or in Economics. The area of optimization has received enormous attention in recent years, primarily because of the rapid progress in computer technology, including the development and availability of user-friendly software, high speed and parallel processors.

The optimization toolbox of MATLAB and many other commercial software like this has given a new dimension to it.

Global Optomization

Optimization is essentially the art, science and mathematics of choosing the best among a given set of finite or infinite alternatives. The task of global optimization is to find a solution in the solution set for which the objective function obtains its smallest value, the global minimum. Global optimization thus aims at determining not just "a local minimum" but "the smallest local minimum" with respect to the solution set.

Extending the class of functions to include multi-modal functions makes the global optimization problem unsolvable in general. In order to be solvable some smoothness condition on the function in addition to continuity must be known.

Methods Available

Many methods were developed in the late 1960s that performs well in optimization are

- **Genetic Programming (GP)**
  Genetic Programming[9] is a related technique popularized by Koza [9] in which computer programs, rather than function parameters, are optimized. Genetic programming often uses tree-based internal data-structure to represent the computer programs for adaptation instead of the list structures typical of genetic algorithms.
- **Interactive Genetic Algorithm (IGA)**
  Interactive Genetic Algorithm are genetic algorithms that use human evaluation. They are usually applied to domains where it is hard to design a computational fitness function, for example, evolving images, music, artistic designs and forms to fit users' aesthetic preference.
- **Simulated Annealing (SA)**
  Simulated Annealing[8] is a related global optimization technique that traverses the search space by testing random mutations on an individual solution proposed by Kirkpatrick and Cerny, [8]. A mutation that increases fitness is always accepted. A mutation that lowers fitness is accepted probabilistically based on the difference in fitness and a decreasing temperature parameter. In SA parlance, one speaks of seeking the lowest energy instead of the maximum fitness. SA can also be used within a standard GA algorithm by starting with a relatively high rate of mutation and decreasing it over time along a given schedule.
- **Tabu Search (TS)**
  Tabu Search is similar to Simulated Annealing in that both traverse the solution space by testing mutations of an individual solution was proposed by Glover [4]. While simulated annealing generates only one mutated solution, tabu search generates many mutated solutions and moves to the solution with the lowest energy of those generated. In order to prevent cycling and encourage greater movement through the solution space, a tabu list is maintained of partial or complete solutions. It is forbidden to move to a solution that contains elements of the tabu list, which is updated as the solution traverses the solution space.
- **Ant Colony Optimization (ACO)**

Ant Colony Optimization uses many ants (or agents) to traverse the solution space and find locally productive areas was proposed by Dorigo in his Ph. D. Thesis[2]. While usually inferior to genetic algorithms and other forms of local search, it is able to produce results in problems where no global or up-to-date perspective can be obtained, and thus the other methods cannot be applied.

- **Memetic Algorithm (MA)**
  Memetic Algorithm also called hybrid genetic algorithm among others, is a relatively new evolutionary method where local search is applied during the evolutionary cycle. The idea of memetic algorithms comes from memes, which–unlike genes–can adapt themselves. In some problem areas they are shown to be more efficient than traditional evolutionary algorithms.
- **External Optimization (EO)**
  Unlike GAs, which work with a population of candidate solutions, EO evolves a single solution and makes Local modifications to the worst components. This requires that a suitable representation be selected which permits individual solution components to be assigned a quality measure ("fitness"). The governing principle behind this algorithm is that of emergent improvement through selectively removing low-quality components and replacing them with a randomly selected component. This is decidedly at odds with a GA that selects good solutions in an attempt to make better solutions.

Genetic Algorithm

The GA was invented by Holland [6] at the University of Michigan in 1975, and subsequently it has been made widely popular by Goldburg [5] , at the University of Illionis. The original GA and its many variants, collectively known as genetic algorithms, are computational procedure that mimics the natural process of evolution.

GAs work by evolving a population of individuals over a number of generations. A fit value is assigned to each individual in the population, where the fitness computation depends on the application. For each generation, individuals are selected from the population for reproduction, the individuals are crosses to generate the new individuals, and new individuals are mutated with some low mutation probability. The new individuals may completely replace the old individuals in the population, with a distinct generations evolved [5].

Outline of Basic Genetic Algorithm

Step1 Choose a coding to represent problem parameters (Our program uses the binary coding of the population), a selection operator, a crossover operator, and a mutation operator. Choose population size, n, crossover probability, p_c and mutation
probability p_m . Initialize a random population of strings of size l. Choose a maximum allowable generation number $t_{max.}$ Set t=o.
Step 2 Evaluate each string in the population. The program we have used uses the evolution via survival of fittest.
Step 3 If $t > t_{max}$ or other termination criteria satisfied, Terminate.
Step 4 Perform reproduction on the population. The selection scheme we have used is the tournament selection with a shuffling technique for choosing random pairs for mating.

Step 5 Perform crossover on random pairs of strings, we have used single point crossover and there is option for uniform crossover.
Step 6 Perform mutation on every string, the program the jump mutation and creep mutation. Niching (Sharing) is also done .
Step 7 Evaluate strings in the new population. Set *t=t+1* and go to step 3.

Repulsive Particle Swarm Optimization

PSO is a form of swarm intelligence [7]. PSO can be easily implemented and it is computationally inexpensive, since its memory requirements are low [3]. Moreover it doses not require gradient information of the objective function under consideration, but only its value. This is modelled by particles in multidimensional space that have a position and a velocity. These particles are flying through hyperspace and have two essential reasoning capabilities: their memory of their own best position and knowledge of the swarm's best. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. There are two main ways this communication is done: a global best that is known to all "neighborhood" bests where each particle only communicates with a subset of the swarm about best positions

The repulsive particle swarm optimization is variant of Particle swarm optimization and belongs to a stochastic evolutionary global optimizers. There are several different realizations of RPSO. Common to all these realization is the repulsion between the particles. This can prevent the swarm trapped in local minima, which would cause a premature convergence and would lead the optimization algorithm to fail to find the global optimum.

The modification of basic PSO scheme is to modify the velocity update formula when the swarm diversity becomes less then the a fixed value (i.e. d_{low}) The velocity is updated by the formula

$$v_{i+1} = \omega v_i + \alpha r_1(\hat{x}_i - x_i) + \omega\beta r_2(\hat{x}_{hi} - x_i) + \omega\gamma r_3 z$$

$$x_{i+1} = x_i + v_{i+1}$$

where,

- x is the position and v is the velocity of the individual particle. The subscripts i and $i+1$ stand for the recent and the next (future) iterations, respectively.
- r_1, r_2, r_3 are random numbers, $\in[0,1]$; α, β, γ are constants
- ω is inertia weight, $\in[0.01, 0.7]$; z is a random velocity vector
- $\hat{x}$ is the best position of a particle; x_h is best position of a randomly chosen other particle from within the swarm [11][12].

Test Functions

The objective of this paper is to present a comparative study of the performance of the Genetic algorithm and Repulsive particle swarm method on the following functions. These functions are difficult in nature. We present these functions in details[10]. A graphical presentation is also given [10].

I. Weierstrass function: The Weierstrass function [in its original form, $f(x)=\sum_{k=0}^{\infty} a^k \cos(b^k x)$ while b is an odd integer, $0<a<1;\ ab>(1+3\pi/2)$] is one of the most notorious functions (with almost fractal surface) that changed the course of history of mathematics. Weierstrass proved that this function is *throughout continuous but nowhere differentiable*. In its altered form this function in m ($m\geq 1$) variables with search domain $[-0.5\leq x_i\leq 0.5]$; $(i=1,2,...,m)$ and the minimum $f(x^*)=0$ for $x^*=(0,\ 0,...,0);\ a=0.5;\ b=3;\ \mathbb{k}=20$, is given as

$$f(x)=\sum_{i=1}^{m}\sum_{k=0}^{\mathbb{k}}[a^k\cos(2\pi b^k(x_i+0.5))]-m\sum_{k=0}^{\mathbb{k}}[a^k\cos(2\pi b^k 0.5)];\ x_i\in[-0.5,\ 0.5];\ i=1,2,...,m$$

II Zettle function: In the search domain $x_1,x_2\in[-5,5]$ this function is defined as follows and has $f_{\min}(-0.0299,0)=-0.003791$

$$f(x)=(x_1^2+x_2^2-2x_1)^2+0.25x_1$$

III Zero-sum Function #7: Defined in the domain $x\in[-10,10]$ this function (in $m\geq 2$) has f(x)=0 if $\sum_{i=1}^{m}x_i=0$. Otherwise $f(x)=1+\left(10000\left|\sum_{i=1}^{m}x_i\right|\right)^{0.5}$. This function has innumerable many minima but it is extremely difficult to obtain any of them. Larger is the value of m (dimension), it becomes more difficult to optimize the function.

IV Dixon & Price function: This function is in m ($m\geq 2$) variables with search domain $[-10\leq x_i\leq 10]$; $(i=1,2,...,m)$ and the minimum $f(x^*)=0$. It is given as

$$f(x)=(x_1-1)^2+\sum_{i=2}^{m}i(2x_i^2-x_{i-1})^2$$

V Trid function: This function is in m ($m\geq 2$) variables with search domain $[-m^2\leq x_i\leq m^2]$; $(i=1,2,...,m)$.The Trid function is given as

$$f(x)=\sum_{i=1}^{m}(x_i-1)^2-\sum_{i=2}^{m}x_i x_{i-1}$$

Optimal values of Trid function of different dimensions (m)																
m	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	$f(x^*)$
15	15	28	39	48	55	60	63	64	63	60	55	48	39	28	15	-665
10	10	18	24	28	30	30	28	24	18	10						-210
6	6	10	12	12	10	6										-50

The values of $f(x^*)$ and those of x^* at different m are given in the table above. The pattern observed in the values taken on by decision variables is interesting.

VI Levy function No. 3: It is a 2-variable ($x_{1,2}$) multi-modal function with search domain $[-10 \le x_i \le 10]$. It has some 760 local minima and 18 global minima in this search domain. Its global minimum is $f(x^*) = -176.542$.

$$f(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \sum_{i=1}^{5} i\cos[(i+1)x_2 + i]$$

VII Beale function: A $2-$variable ($m=2$) function with search domain $[-4.5 \le x_i \le 4.5]$; ($i=1,2$) given as

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

is called the Beale function. It is a uni-modal function with the global minimum $f(x^*) = 0$ for $x^* = (3,\ 0.5)$. The two values, (3, 0.5), have a definite relationship with the constants in the function. The constant in the first term (i.e. 1.5) is equal to $3 - 3(0.5)$. The constant in the second term (i.e. 2.25) is equal to $3 - 3(0.5)^2$. The constant in the third term (i.e. 2.625) is equal to $3 - 3(0.5)^3$. Let us now call these parameters (a, b). Then the constant in the first term is $a - ab^1$, the constant in the second term is $a - ab^2$ and the constant in the third term is $a - ab^3$. For any positive a and positive fractional b $(0 < b < 1)$ the function may be generalized. The optimal solution would be (a, b) and $f(x^*) = 0$.

VIII Booth Function: A $2-$variable ($m=2$) function with search domain $[-10 \le x_i \le 10]$; ($i=1,2$) given as

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

This function is multimodal with the global minimum $f(x^*) = 0$ at $x^* = (1,\ 3)$.

IX Easom function: This function is in 2 variables ($m=2$) with search domain $[-100 \le x_i \le 100]$; ($i=1,2$) and $f(x^*) = -1$ at $x^* = (\pi,\ \pi)$. It is given as

$$f(x) = -\cos(x_1)\cos(x_2)\exp[-(x_1 - \pi)^2 - (x_2 - \pi)^2].$$

X Himmelblau function: It is a 2-variable ($m=2$) function with search domain $[-6 \le x_i \le 6]$; ($i=1,2$) and 4 global minima $f(x^*) = 0$, one each in the four Cartesian quadrants. The optimal values of x are: (3,2), (-2.805, 3.131), (-3.779, -3.283) and (3.584, -1.848). The function is written as:

$$f(x) = (x_1 + x_2^2 - 7)^2 + (x_1^2 + x_2 - 11)^2$$

The modified Himmelblau function has only one global optimum $f(x^*) = 0$ at $x^* = (3,2)$. This (modified) function is given as

$$f(x) = (x_1 + x_2^2 - 7)^2 + (x_1^2 + x_2 - 11)^2 + 0.1[(x_1 - 3)^2 + (x_2 - 2)^2]$$

Experiments

A Algorithms used for the comparative study were Genetic Algorithm, Improved-Repulsive Particle swarm Optimization & Simulated Annealing. For all algorithms the dimensions were set manually , thus based on few preliminary experiments.

B Genetic algorithms: We have used and input file to pass the different parameters i.e. npopsiz=5, pcross=.9d0, npsibl1=(2^{*N} N= powers of 2) pmutate=0.02d0 and maxgen=200. Another params.f was included in the main program having three parameters population size=200, nchrommax=60 and nparamax=10. other two parameters are adjustable according to the dimensions of the problems.

C RPSO setting: RPSO have several parameters population size=40, In most of the cases n=30 works fine. Its value can be increased up to 50 to 100. A randomly chosen neighbors NN=15. The maximum no of decision variables MX=100, Number of iteration was set 1000.

Results

SN	Functions	Dim	GA	RPS	True Value
1	Weierstrass function	2	**0.000000**	0.029900	0.7513280664284E-08
2	Zettle function	5	**0.000000**	0.00379	0.3791236557044E-02
3	Zero-sum Function #7	2	0.000000	**-1.00000**	-1.00000
4	Dixon & Price function	5	0.600000	**0.000000**	0.7368500368739E-08
5	Trid function	5	-2.000003	**-30**	-29.999999530
6	Levy function No. 3	2	17.15224	**-176. 54179**	-176.5417931343
7	Beale function:	2	5.45315	**0.000000**	0.1080137747535E-09
8	Booth Function	2	20.99958	**0.000000**	0.4368455356320E-09
9	Easom function	2	**1.000000**	**1.0000000**	0.9539719592261
10	Himmelblau function	2	**0.0000000**	**0.000000**	0.1476885118888E-08

Conclusion

Our program of Genetic algorithm has given a good result for the functions like Weierastrass function, Zettle function and Easom function and fails for other functions considered for experiments where the Repulsive particle Swarm (RPS) have failed in these functions except the Easom function and Himmelblau function where both finds out the solution.

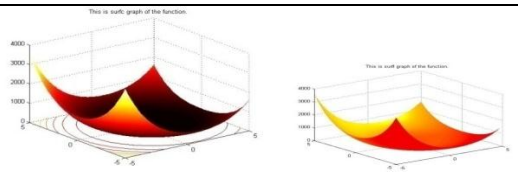
Whereas Repulsive Particle Swarm(RPS) has performed better for Dixon Price function, Trig function, Levy #3 function, Zero sum function 7 ,Levy function 3, Bealy function, Booth function then Genetic Algorithm(GA). The results shown in the above table by bold where the methods has outperformed the other. This shows that no algorithm is able to absolutely outperform the other.

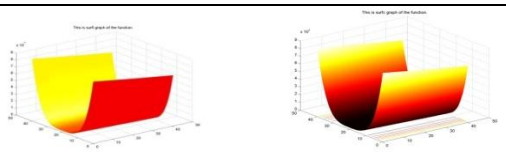
Acknoledgements
David L. Carrol l, from CU, South Wright Street Extended Urbana for using the GA program with a modification. And S. K. Mishra, department of Economics, NEHU Shillong India for conceiving the idea and the computer program.

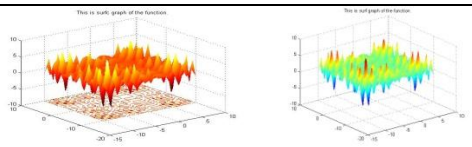
Appendix

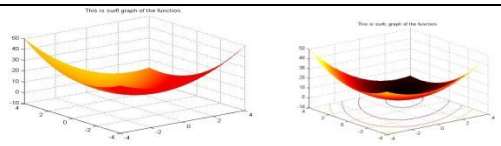
Here we present the visual representation of the functions considered for comparative study. MATLAB 7.1 SP2 have been used to redraw the graph of the functions using mesh, meshc,surf ans surfc.

Name of Functions	Visual Presentation	MATLAB Code

Zettle Function:		`r=-5:0.1:5;`
		`[X,Y]=meshgrid(r,r);`
		`Z=(X.^2+Y.^2-2*X).^2+0.25*X;`

Dixon & Price function:		*`% Dixon Function`* *`r=-10:.5:10;`* *`[X,Y]=meshgrid(r,r);`* *`Z=(X-1).^2+2*(2*Y.^2-X).^2;`*

Levy function#3		`r=-10.1:10;`
		`[X,Y]=meshgrid(r,r);`
		`Z=(cos(2.*X+1)+2.*cos(3.*X+2)).*(cos(2.*Y+1)+2.*cos(3.*Y+2));`

Trid Function:		`r=-4:.1:4;`
		`[X,Y]=meshgrid(r,r);`
		`Z=((X-1).^2+(Y-1).^2)-Y.*X;`

Code to draw the graph:

```
% Graph Function
%This is mesh graph of the function.
figure(1)
mesh(X,Y,Z)
```

```
title('This is mesh graph of the function.')
%This is surf graph of the function
figure(2)
surf(X,Y,Z,'facecolor','interp','Edgecolor','none','facelighti
ng','phong')
axis('on')
title('This is surf graph of the function.')
%This is a surface countor of the function.
figure(3)
surfc(X,Y,Z,'facecolor','interp','Edgecolor','interp','facelig
hting','phong')
colormap hot
axis('on')
title('This is surfc graph of the function.')
%This is surface illuminated  graph of the function.
figure(4)
surfl(X,Y,Z)
title('This is surfl graph of the function.')
shading interp
colormap hot
```

Figures of the above test functions drawn using MATLAB. The credit of some of the functions goes to AR Hedar, Dept. of Computer Science, Faculty of Computer & Information Sciences, Assiut University, Egypt.

http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm

References

[1] Cerny, V., 1985. "*Thermodynamical Approach to the Traveling Salesman* Problem: An Efficient Simulation Algorithm", J. Opt. Theory Appl., 45,1,41-51.

[2] Dorigo M., 1992, "Optimization, Learning and Natural Algorithm", Ph.D. Thesis, Dip. Electronica & Informazione, Politechnico de Milano, Italy.

[3] Eberhart RC, Simpson P and Dobbins R ,1996, "Computational Intelligence PC Tools"' Academic Press.

[4] Glover F. 1986'. "Future paths for *integer Programming and links to Aritificial Intelligence*", Computers and Operations Research, 5:533-549.

[5] Goldburg, DE, 1989 "*Genetic Algorithm in search, Optimization, and Machine Learning"*, Reading, MA Addition-Wesley.

[6] Holland JH, 1975, "*Adaptation in Natural and Artificial Systems"*, Ann Arbor, MI: University of Michigan Press.

[7] Kennedy J and Eberhart RC, 2001, "*Swarm Intelligence",* Morgan Kaufmann Publishers.

[8] Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi, M.P, 1983,. "*Optimization by Simulated Annealing*", Science, 220, 4598, 671-680.

[9] Koza, John , 1992.,"*Genetic Programming: On the Programming of Computers by Means of Natural Selection"*, MIT Press. ISBN 0-262-11170-5.

[10] Liang J. J., Suganthan P.N. and Deb K., 2005, "*Novel Composition test functions for numerical global optimization,"* Proc. Of IEEE International Swarm Intelligence Symposium, pp. 68-75.

[11] Liang J.J. and Suganthan P.N., 2005 ,"*Dynamic Multi-swarm Particle Swarm Optimizer, "* Proc. of IEEE International Swarm Intelligence Symposium pp. 124-129.

[12] Mishra, S. K.(2006): Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method SSRN: http://ssrn.com/abstract=926132.