Journal of
Computer Science & Systems Biology

Singh et al., J Comput Sci Syst Biol 2016, 9:2
DOI: 10.4172/jcsb.1000220

Review Article Open Access

A Hybridized Chemotactic Genetic Algorithm for Optimization

Prabhash Kr Singh[1]*, Shyamal Kr Mondal[2] and Manoranjan Maiti[2]

[1]Department of Computer Science, Vidyasagar University, Midnapore, West Bengal, India
[2]Department of Applied Mathematics with Oceanology and Computer Programming, Vidyasagar University, Midnapore, West Bengal, India

Abstract

This paper presents a new hybridized genetic algorithm for optimization of a function with several variables. To construct the proposed algorithm, a new operator by the name of chemotactic operator has been constructed using the chemotaxis behaviour of the *E. coli* bacteria. Then this integrated genetic algorithm has been used to obtain minimum optimal of a single objective function. Finally, some numeric simulations have been carried out to show the effectiveness of the proposed algorithm comparing the results of simple genetic algorithm using some benchmark functions.

Keywords: Genetic algorithm; Chemotaxis; Swimming; Optimization; Real coded genetic algorithm; Benchmark functions

Introduction

In last forty years lots of emphasize has been made to solve problems with the help of algorithms that rely on analogies to natural process [1-6]. Different algorithms and its hybridization evolved to give the best optimal point of a problem. However out of many approaches, the approach based on Genetic Algorithm have found great attention from the researchers all around the globe to deal with the optimization problem that have di cult dealing with conventional problem solving techniques [7-9]. In recent period, many ideas taken from the biological activity of an organism have contributed towards the development of the evolutionary algorithm. Scientist and researchers of recent time have been drawing inspiration from nature to tackle the complex search problems of the real world. One such inspiration can be drawn from the fact that *E. coli* bacteria moves taking small steps in search of nutrients, called chemotaxis. Lots of researchers have studied the behavior of this bacteria and emphasize that the movement of the bacteria is controlled by the run and tumble. *E. coli* senses chemoeffector gradients in temporal fashion by comparing current concentrations to those encountered over the past few seconds of travel.

In this paper, we present a hybridized chemotactic genetic algorithm by integrating the chemotaxis behaviour of the *E. coli* bacteria in genetic algorithm which shows an improved result when tested on benchmark objective functions. Each chromosome of the population in genetic algorithm points to one discrete point. On applying these chromosomes to the objective function we try to find the required optimal value and then these chromosomes are passed through crossover and mutation to obtain another set of good chromosome. If the positions of the chromosome swim in any vector direction in search of some good point from the present, a significant improvement in the result could be obtained.

Preliminaries

An overview of *E. coli* chemotaxis

Escherichia coli (*E. coli*) are bacteria which normally live in the intestines of living organisms. Most *E. coli* bacteria are harmless and actually are an important part of a healthy human intestinal tract. It is equipped with a set of rotary motors only 45 nm in diameter. Each motor drives a long, thin, helical lament that extends several cell body lengths out into the external medium. The assemblage of the motor and lament is called a flagellum [10]. The movement of the bacteria towards or away from chemicals called chemotaxis, is a universal attribute of motile cells and organisms. *E. coli* cells swim toward potentially non-noxious amino acids (serine and aspartic acid), sugars (maltose, ribose, galactose, glucose), dipeptides, pyrimidines and electron accep-tors (oxygen, nitrate, fumarate) but runs away from potetially noxious chemicals, such as alcohols and fatty acids. Now, in isotropic chemical environments, *E. coli* swim in a random walk pattern produced by alternating episodes of counter-clockwise (CCW) and clockwise (CW) flagellar rotation (Figure 1, left panel). In an attractant or repellent gradient, the cells monitor chemoeffector concentration changes as they move about and use that information to modulate the probability of the next tumbling event (Figure 1, right panel).

When the cell's motors rotate CCW, the flagellar filaments form a trailing bundle that pushes the cell forward. When one or more of the flagellar motors reverses to CW rotation, that filament undergoes a shape change (flowing to the torque reversal) that disrupts the bundle. Until all motors once again turn in the CCW direction, the filaments act independently to push and pull the cell in a chaotic tumbling motion. Tumbling episodes enable the cell to try new, randomly determined swimming directions. Sensory information suppresses tumbling whenever the cell happens to head in a favorable direction. The cells

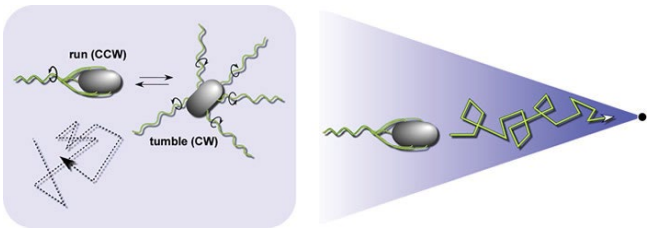

Figure 1: Run and Tumble of *E. coli* Bacteria.

***Corresponding author:** Prabhash Kr. Singh, Department of Computer Science, Vidyasagar University, Midnapore-721102, West Bengal, India, E-mail: singhg11@gmail.com

Received December 29, 2015; **Accepted** February 26, 2016; **Published** February 29, 2016

Citation: Singh PK, Mondal SK, Maiti M (2016) A Hybridized Chemotactic Genetic Algorithm for Optimization. J Comput Sci Syst Biol 9: 045-050. doi:10.4172/jcsb.1000220

cannot head directly upgradient because they are frequently knocked o course by Brownian motion. These locomotor responses extend runs that take the cells in favorable directions (toward attractants and away from repellents), resulting in net movement toward preferred environments. These behavioral responses to the environmental stimuli allow the bacteria to find optimal conditions for growth around their surroundings.

Genetic algorithm

It is an algorithm mainly based on the ideas and the techniques from genetic evolutionary theory [11]. It follows the principle of the survival of the fittest and generates the next population by going through many operations, with each individual representing a possible solution. This algorithm has been applied to various domains such as inventory control, image processing, pipeline control, information retrieval etc. The basic operations that it performs are selection, crossover and mutation. A typical genetic algorithm generates some random populations which are being passed through a fitness function to judge whether it will contribute to the next generation of solutions. The chromosome which has greater probability of contribution to the next population gets selected. As in the biological process the sexual reproduction takes place which exchanges and reorders the chromosome to produce the offspring. The same is exhibited to the newly generated population by applying crossover operators to obtain child solution by mixing one or more parent solution. After the reproduction, the mutation operator is applied to the chromosome to bring about a variety of different chromosome in the population. The probability of mutation is generally low. This procedure is again iterated for a finite number of times to find the optimal parameters and its corresponding objective function value. The algorithm will terminate either when the maximum number of generations for iteration has been reached or desired fitness value of the population has been achieved. Genetic algorithm continually exploits new and better solutions without any pre-assumptions, such as continuity and unimodality. The application of crossover and mutation in almost all iteration avoids premature convergence of the problem. The different operators used in Simple Genetic Algorithm (SGA) are discussed in the following sections.

Selection: This operator is use to carry good chromosomes with better fitness from the current generation to the next generation in an assumption that the selected chromosomes when passes through crossover and mutation will have better result than the current generation. A careful strategy has to be negotiated so that most of the better chromosomes could be forwarded to the next generation and the worse could be left out. There are several selection schemes, such as roulette wheel selection, ranking selection, tournament selection, etc.

Crossover: It is a technique to randomly select two parents for mating from the population of chromosomes obtained from selection operator to produce a child called offspring. The obtained offspring is generally believes to be better that both the parents if it takes good characteristics from them. This crossover operator is controlled by the crossover probability p_c. In the literature, lots of crossover techniques are present for real coded genetic algorithm, such as uniform crossover, arithmetic crossover, simulated binary crossover and many others.

Mutation: The mutation operator plays a very vital role in Genetic Algorithm. It helps to maintain diversification by incorporating some uniqueness between the chromosomes with some probability p_m. It explores the unexplored points within the domain of the variables of the chromosome so that the premature convergence does not take place in sub optimal solutions. The different mutation techniques available in the literature are Makinen, Periaux and Toivanen mutation, Dynamic mutation, Power Mutation, Uniform Mutation, Boundary Mutation etc.

The Proposed Hybridized Chemo-Tactic Genetic Algorithm (HCGA)

In this section a new hybridized Genetic Algorithm has been developed in which the chemotaxis beahavior of *E. coli* bacteria has been captured as an operator in genetic algorithm by which searching procedure has been defined rather than simple genetic algorithm. In Simple Genetic Algorithm (SGA), each individual is a chromosome which has been obtained after encoding the variables present within the search space. Each variable represents a discrete point within its specified domain space. As it gets evolved or passes through different genetic operators such as crossover and mutation, it points to a position in the search domain which is at some significant distance from the previous points. The problem in this technique is that at most of the time it fails to traverse the very nearby points of the variables before using the genetic operators which otherwise could actually lead to attain a very accurate result of the considered optimization problem. This traversing the nearby points of the variables has been mimicked from the chemotaxis behaviour of *E. coli* bacteria which swims through the points for some swimming length distance in order to find the optimal position and then it tumbles itself to sense the next random direction of swimming through the search space to find the better optimal points.

Figure 2 depicts the flowchart of the proposed algorithm. Here, in the proposed hybridization we allow the initial chromosomes to be generated randomly in the population. Then the chromosomes are allowed to swim from the current position p to position $(p+\Delta)$, where Δ indicates a very small displacement in the position of the variables in the random direction. This process of swimming will continue till it finds favourable condition but for limited period of time. The swimming of the chromosome helps to traverse the very nearby points in the random direction. This helps to get more optimum value of the optimization problem. Also, this feature helps to climb to good points which favour the need to optimization problem and at same time it avoids the points which are not favourable. This total process of swimming and tumbling for each chromosome will iterate as long as we get better position within the search domain of the variables or till maximum number of chemo-tactic step have been attained, whichever is earlier. The different components used in HCGA are described below:

(a) **Chromosome representation:** In this paper, the real number representation scheme is used. Here, the structure of a chromosome is a K dimensional real vector, $X = (x_1, x_2, \ldots, x_K)$, where $x_1, x_2, \ldots, x_K$ represent different decision variables of the problem.

(b) **Initialization:** Using the chromosome representation, a population set of N solutions (chromosomes) $X_1, X_2, \ldots, X_N$ are randomly generated satisfying the upper and lower bound of the given domain space of the problems.

(c) **Fitness value:** All the chromosomes in the population are evaluated using the fitness function of the given problem. Here, the minimization of the value of an objective function $f(X)$, due to the solution X, is taken as fitness of X.

(d) **Selection process to create mating pool:** Here, the binary tournament selection process of size 2 is used for all the benchmark problems. In this process, randomly choose any two chromosomes from the current population and select the best chromosome among

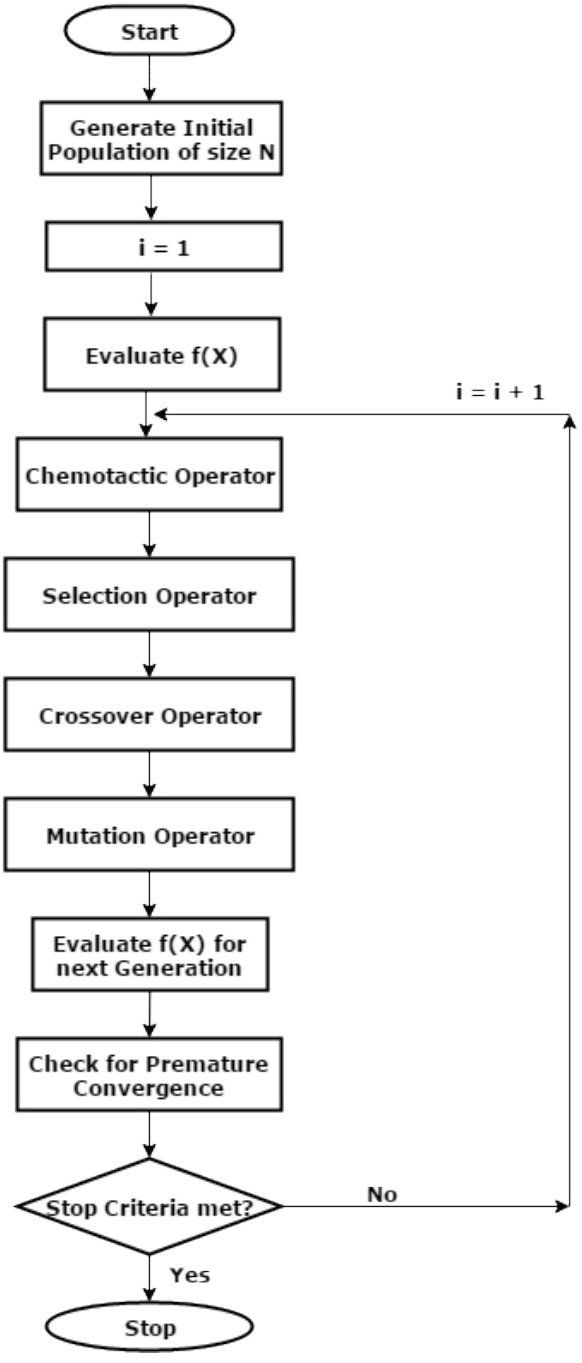

Figure 2: Flowchart of HCGA.

them for creation of mating pool. This process is repeated N times to obtain the new N solution set for the next generation.

(e) **Proposed new chemotactic operator:** The *E. coli* bacteria always try to nd the optimal point with respect to its surroundings. It searches the optimal point through swimming and tumbling. The movement of the bacteria can be modeled mathematically and can be applied as new operator to the existing genetic algorithm for obtaining better optimal point. Consider the initial position of the bacteria as x_1, x_2, ..., x_K in K dimensional real vector as an analogy to the chromosome in the problem space. Apply the following step for every chemotactic movement:

Step-1: for j= 1, 2,...., N_c, N_c is the number of chemotactic steps

Step-2: Displace each position x_p, p=1,2,...,K in any random direction of a chromosome X_i, i=1,2,...,N such that:

$$X_i(j+1) = x_p + \psi \frac{\Delta x_p}{\sqrt{\sum_{p=1}^{K} \Delta(x_p)}} \tag{1}$$

where, $\Delta(x)$ lies in [-1, 1] ψ and represents the step size of the chromosome in the random direction.

Step-3: $swim$=0

Step-4: while ($swim$<N_s), N_s is the number of swim steps

Step-5: $swim$=$swim$ + 1

Step-6: if ($f(X_i(j+1)) < f(X_i(j))$)

Step-7:
$$X_i(j+1) = X_i(j+1) + \psi \frac{\Delta x_p}{\sqrt{\sum_{p=1}^{K} \Delta(x_p)}} \tag{2}$$

Step-8: else $swim$=N_s

Step -9: end while

Step-10: end for

(f) **Crossover:** After the chemotactic process, for each solution of the population at the next generation, generate a random number r from the range [0....1]. If r<p_c then the solution is taken for crossover. In this paper, the simulated binary crossover (SBX) [12] is chosen as crossover operator for each selected pair of coupled solutions X_1, X_2 to be replace by their offspring's X'_1 and X'_2.

$$\begin{aligned} X'_1 &= 0.5\left[(1-\beta_k)X_1 + (1+\beta_k)X_2\right] \\ X'_2 &= 0.5\left[(1+\beta_k)X_1 + (1-\beta_k)X_2\right] \end{aligned} \tag{3}$$

Here $\beta_k(\geq 0)$ is a sample from a random number generator having the density

$$p(\beta) = \begin{cases} 0.5(\eta_c+1)\beta^{\eta_c}, & \text{if } 0 \leq \beta \leq 1 \\ 0.5(\eta_c+1)\beta^{\frac{1}{\beta^{\eta_c+2}}}, & \text{otherwise} \end{cases}$$

The distribution index η_c is any non-negative real number. A high value of η_c will have the higher probability of creating near parent solution where as low value of $_c$ gives distant solution to be selected as child.

(g) Mutation: Here, for diversity of the population, the dynamic mutation operator [13] is used in the proposed algorithm:

$$\begin{aligned} x'_k &= x_k + \Delta(k, x_k^u - x_k), \tau = 0 \\ x'_k &= x_k - \Delta(k, x_k - x_k^l), \tau = 1 \end{aligned} \tag{4}$$

where the random constant τ becomes 0 or 1 and $\Delta(k, y)$ is given as

$$\Delta(k, y) = y.r.\left(1 - \frac{k}{T}\right)^b \tag{5}$$

where r is a random 0 or 1 and T is the maximum number of generation given.

(h) Premature convergence: The premature convergence in genetic algorithm occurs due to loss of diversity in the population which avoids creation of better o spring than the parent. To avoid the premature convergence in the proposed algorithm, at the end of every generation the following is checked

$$|\alpha_c - \alpha_g| < \varepsilon \quad (6)$$

where α_c is the best chromosome of the current generation, α_g is the best chromosome found till the current generation. If the Eq. (6) holds for consecutive 10 generations then the rate of mutation is significantly increased for subsequent one successive generation.

Therefore, the step-wise procedure of HCGA can be written as:

Step-1: Generate initial population P_1 of size N

Step-2: $i \leftarrow 1$ [i represents the no. of current generation]

Step-3: Obtain the fitness value of the optimization problem

Step-4: Apply chemotactic operator

Step-5: Apply selection operator

Step-6: Apply crossover operator

Step-7: Apply mutation operator

Step-8: Compute the fitness value for next generation

Step-9: Check for premature convergence:

Step-10: Set $i \leftarrow 1 + 1$

Step-11: If termination condition does not hold, go to Step-4

Step-12: End

Numerical Illustration

To compare between the proposed Hybridized Chemotactic Genetic Algorithm (HCGA) and traditional Simple Genetic Algorithm (SGA), some test functions are collected as given in the Table 1. Each test function considered has a different behaviour in its search domain. For a uniform testing environment all the parameters used as in Table 2 kept same for all the test functions. To avoid premature convergence uniform mutation with increased probability is applied when the difference between the best value of the current generation and best optimal value till current generation is less than some value for consecutive 10 generations. The number of variables for all the test function is set to be 10 except for Booth's function. Both the algorithms are compared and analyzed with 10 independent runs. For a particular test problem, a run is said to be a successful run if the best objective function value found in that run lies within 1% accuracy of the best known objective function value of that problem. The maximum numbers of generations are also fixed to be 2000 for both the algorithm. All the algorithms are implemented in *C* environment.

After considering the test functions, the average optimal function value of the 10 independent runs are shown in Table 3 and in Table 4, an average optimal value of 10 independent runs at different

Function	Mathematical representation	Search Domain	Global Optima
Rastrigin	$f_6(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-5.12,5.12]	$f_1(0)=0$
Ackley	$f_2(x) = -20 * \exp\left(0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-30,30]	$f_1(0)=0$
Axisparallel Hyperellipsoid	$f_3(x) = \sum_{i=1}^{n} i x_i^2$	[-5.12,5.12]	$f_3(0)=0$
Booth	$f_4(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	[-10,10]	$f_4(0)=0$
Cosine Mixture	$f_5(x) = 0.1n + \sum_{i=1}^{n}[x_i^2 - 0.1\cos(5\pi x_i)$	[-1,1]	$f_5(0)=0$
Rosenbrock	$f_6(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-2.048,2.047]	$f_6(0)=0$

Table 1: Benchmark function used.

Algorithm	Parameters
GA	p_c=0.9, p_m=0.01, N=50
HCGA	N_c=100, N_s=4, ψ=0.001, p_c=0.9, p_m=0.01, N=50

Table 2: Parameters used for Benchmark function.

Test Functions	SGA		HCGA	
	Optimal Value	Success Rate (%)	Optimal Value	Success Rate (%)
f_1	0.233003	70	0	100
f_2	0.126791	30	0.030271	60
f_3	0	100	0	100
f_4	0.000054	100	0	100
f_5	0	100	0	100
f_6	6.735035	0	5.556376	20

Table 3: Comparison between both the algorithms on test functions.

generations are shown. From Table 3 it can be analyzed that the proposed method gives better value and has better success rate on most of the test functions and in some cases it gives same optimal value with 100% success rate as of SGA. Also, from the result obtained in Table 4 we can clearly state that the HCGA performs better at each generation of the test function than the SGA. Furthermore, to provide a fair basis comparison, we recorded the best, median and the worst solutions along with the average standard deviation of the 10 independent runs of both the algorithm. From the results shown in Table 5, it is clear that the proposed algorithm has slight edge on Simple Genetic Algorithm. Figures 3-8 compare the convergence of both the algorithm SGA and HCGA for the test functions f_3 to f_7 respectively. As evident from the Figures 3-8 that the both the algorithm have near about same value for initial 70 100 generations but after that in subsequent generation HCGA gains the advantage and performs better. From the given figures, it is quite clear that the proposed algorithm gives better optimal value with greater accuracy for all the used test functions.

Conclusion

This research article integrates the chemotaxis movement of the bacteria with the traditional Genetic Algorithm and presents

Test Function	Algorithm	200	400	600	800	1000	1200	1400	1600	1800	2000
f_1	SGA	5.806156	5.472274	5.273376	4.183591	4.178828	0.742085	0.233003	0.233003	0.233003	0.233003
	HCGA	4.775809	3.681447	3.482409	2.544939	1.88554	0.412678	0.000003	0	0	0
f_2	SGA	0.506244	0.493729	0.493729	0.473954	0.460924	0.433497	0.340405	0.264181	0.126813	0.126791
	HCGA	0.395546	0.346946	0.280493	0.249222	0.239264	0.231398	0.182504	0.176517	0.030277	0.030271
f_3	SGA	0.003083	0	0	0	0	0	0	0	0	0
	HCGA	0.000062	0.000059	0	0	0	0	0	0	0	0
f_4	SGA	0.023897	0.023497	0.023497	0.023497	0.023497	0.023497	0.023497	0.019913	0.006276	0.000054
	HCGA	0.010459	0.003488	0.003052	0.002797	0.001298	0.000541	0.000452	0.000064	0.000018	0.000007
f_5	SGA	0.014785	0.014785	0.014785	0.014784	0.000006	0.000006	0.000006	0	0	0
	HCGA	0	0	0	0	0	0	0	0	0	0
f_6	SGA	9.865311	8.239281	7.397379	7.363624	7.350807	7.338222	7.327611	7.310072	7.13207	6.735035
	HCGA	6.543331	6.421719	6.40028	6.372184	6.242879	6.083023	6.038328	5.920845	5.820096	5.556376

Table 4: Average optimal value of 10 independent runs at different generations.

Test Function	Algorithm	Best	Median	Worst	Standard Deviation	CPU Run Time
f_1	SGA	0	0	1.927159	7.060083	15.2
	HCGA	0	0	0	7.368378	231.6
f_2	SGA	0	0.165332	0.217164	0.381243	15.5
	HCGA	0	0.000002	0.217164	0.39666	208.1
f_3	SGA	0	0	0	6.421538	25.5
	HCGA	0	0	0	6.161951	216.8
f_4	SGA	0	0.000019	0.000211	0.327974	28.9
	HCGA	0	0	0.000181	0.334656	129.7
f_5	SGA	0	0	0	0.117764	23.281
	HCGA	0	0	0	0.114898	296.3
f_6	SGA	3.650792	7.189775	7.383093	31.374885	29.7
	HCGA	0.060152	6.940965	7.342589	31.87433	88

Table 5: Performance comparison of SGA and HCGA on 10 independent runs.

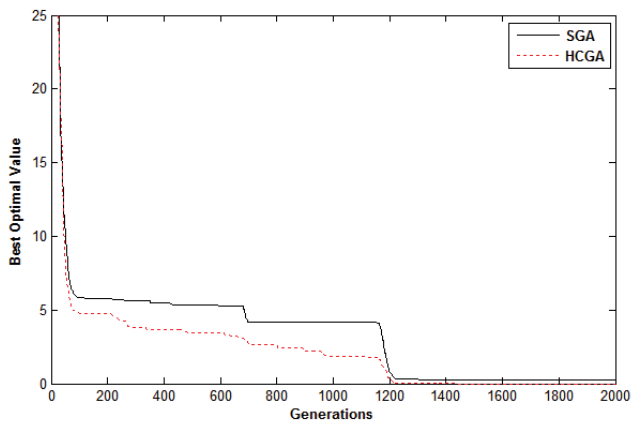

Figure 3: Convergence of SGA and HCGA for test function f_1.

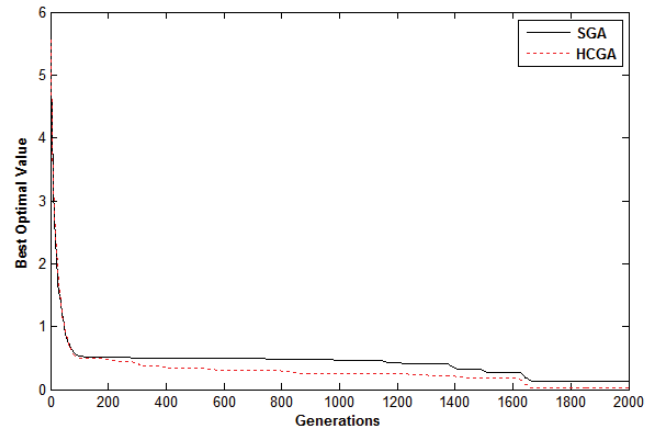

Figure 4: Convergence of SGA and HCGA for test function f_2.

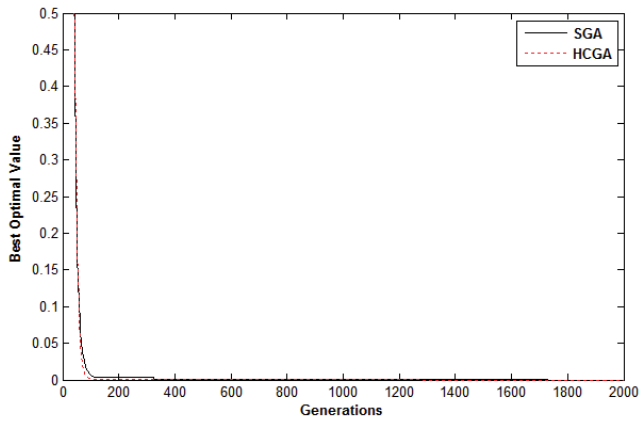

Figure 5: Convergence of SGA and HCGA for test function f_3.

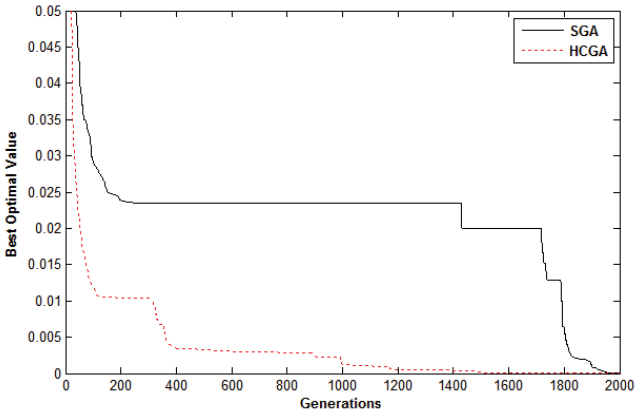

Figure 6: Convergence of SGA and HCGA for test function f_4.

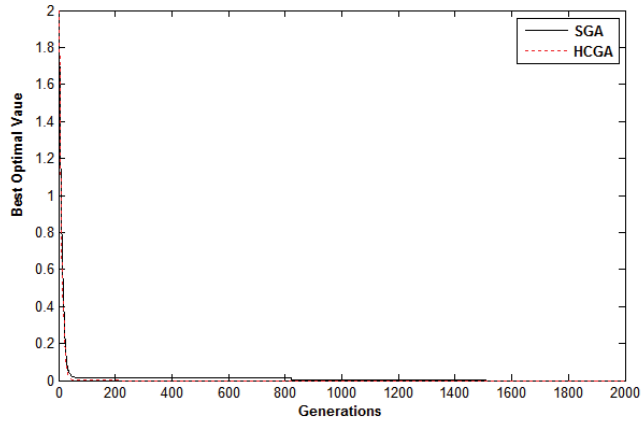

Figure 7: Convergence of SGA and HCGA for test function f_5.

a new hybridized Chemotactic Genetic Algorithm. The behaviour of the new algorithm has been presented and illustrated explicitly using various test functions where we can depict that the proposed algorithm performs better than the SGA. The proposed algorithm can be extended or parameters of the algorithm could be ne tuned to give better optimum result.

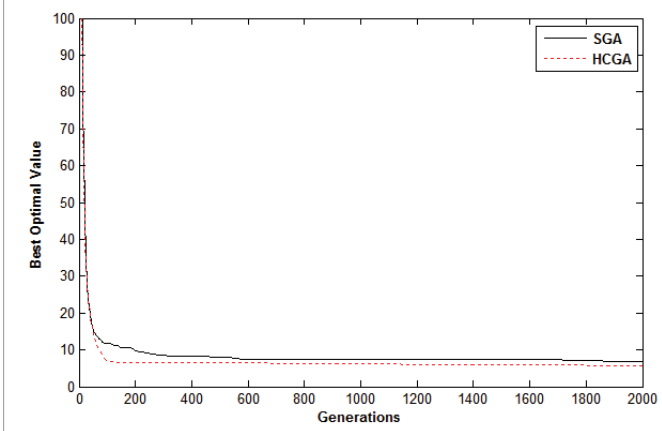

Figure 8: Convergence of SGA and HCGA for test function f_6.

References

1. Goldberg BKDE, Deb K (1989) Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3: 493-530.
2. Srinivas M, Patnaik L (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on System, Man and Cybernetics 24: 656-667.
3. Chun-Liang L, Huai-Wen S (2000) Intelligent control theory in guidance and control system design: An overview. Proceedings of the National Science Council 24: 15-30.
4. Gray GJ, Murray-Smith DJ, Yun L, Sharman KC, Thomas W (1998) Non-linear model structure identification using genetic programming. Control Engineering Practice 6: 1341-1352.
5. Sun X, Gong D, Zhang W (2012) Interactive genetic algorithms with large population and semi-supervised learning. Applied Soft Computing 12: 3004-3013.
6. Yan WWS, Liu X (2010) An improved evaluation method for interactive genetic algorithms and its application in product design. IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), Changsha.
7. Dotoli M, Maione G, Naso D, Turchiano E (2001) Genetic identification of dynamical systems with static nonlinearities. Proceedings IEEE SMC Mountain Workshop Soft Computing Industrial Applications, Blacksburg, VA.
8. Fleming P, Purshouse R (2002) Evolutionary algorithms in control system engineering: A survey. Control Engineering Practice 10: 1223-1241.
9. Fonseca MC, Fleming JP (1998) Multi-objective optimization and multiple constraint handling with evolutionary algorithms, part i: A unified formulation; part ii: Application example. IEEE Transactions on Systems Man and Cybernetics, Part A: Systems and Humans 28: 26-37.
10. Berg H (2005) Motaile behaviour of bacteria. Physics Today 1: 24-29.
11. Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms. IEEE Trans System Man, and Cybernetics 16: 122-128.
12. Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Systems 9: 115-148.
13. Michalewicz Z (1999) Genetic algorithms + data structures = evolution programs. Springer-Verlag, New York.