A Novel Hybrid Self Organizing Migrating Algorithm with Mutation for Global Optimization

Dipti Singh, Seema Agrawal

***Abstract*—This paper presents a Novel Hybrid Self Organizing Migrating Algorithm with Mutation for Global Optimization (M-SOMA). The proposed algorithm includes the hybridization of Self Organizing migrating Algorithm (SOMA) and Non uniform mutation. SOMA is very effective population based algorithm among evolutionary algorithms. Though its convergence is very fast but there are lots of chances to trap in to local optima. As no new points are generated during the search only positions are updated. So to maintain the diversity of the search space and prevent premature convergence it is hybridized with Non Uniform mutation. The proposed algorithm is tested on 15 benchmark unconstrained test problems and its efficiency is compared with SOMA and GA results. On the basis of comparison it is concluded that the presented algorithm shows better performance in terms of function mean best. The graphical results also show that the presented algorithm perform better in terms of efficiency, reliability and accuracy.**

***Index Terms*— Self Organizing Migrating Algorithm, Non-Uniform Mutation, Genetic Algorithm, Global Optimization.**

I. INTRODUCTION

Many real life problems can be formulated as optimization problems arising in almost all areas including robotics, control systems, power systems, biomedical engineering, production engineering and many more. Most of the real life problems come out to be non linear optimization problems and for these problems it is desirable to find the global optimal solution rather than local optimal solution. There are two approaches to solve global non linear optimization problems, one is deterministic and another is stochastic.

Stochastic methods are considered suitable for finding the global optimal solution but sometimes they converge slowly. These are population based algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Self Organizing Migrating Algorithms (SOMA) etc. Among the above mentioned algorithms Self Organizing Migrating Algorithm is relatively new. It is based on the social behavior of a group of individuals. This algorithm was developed by Zelinka and Lampanin in 2000 [1]. This algorithm is inspired by the competitive - cooperative behavior of intelligent creatures solving a common problem. A group of animals such as wolves or other predators may be a good example. If they are looking for food, they usually cooperate and compete so that if one member of this group is successful then the other animals of the group change their trajectories towards the most successful member. If a member of this group is more successful than the previous best one then again all members change their trajectories towards the new successful member. It is repeated until all members meet around one food source.

Manuscript received January 15, 2014.

Dipti Singh, Department of Applied Sciences, Gautam Buddha University, Greater Noida,, India.

Seema Agrawal, , Department of Mathematics, S.S.V. (P.G.) College, Hapur, (C. C. S. University), India.

Like other evolutionary algorithm it also works with a population of solutions. The main feature of this algorithm which makes it distinguished as compared to other algorithms is that no new solutions are created during the search. Instead, only the positions of the solutions are changed during a generation, called a migration loop. Due to which the convergence of SOMA is very fast but it may trap to local optima. It may happen due to the poor balance between exploration and exploitation which are two major characteristics of population based algorithms, in which exploration searches the whole solution space and exploitation converges to the optimal solution. To overcome this difficulty population based algorithms are hybridized with other existing approaches. Kasprzyk and Jasku developed a hybrid genetic – simplex algorithm, wherein a GA finds the most promising solution area over the search space and then the simplex method uses starting points provided by the GA to determine an optimum [2]. Deep and Dipti developed a new hybrid algorithm SOMGA for function optimization and the unique feature of this algorithm is that it is hybridization of binary coded GA and real coded SOMA [3]**.** Khosravi et al. proposes a novel hybrid algorithm that uses the abilities of evolutionary and conventional algorithm simultaneously [4]. Ghatei et al. designed a new hybrid algorithm using PSO and GDA, in this approach, global search character of PSO and local search factor of great deluge algorithm are used based on series [5] . Deep and Thakur proposed a new mutation operator for real coded genetic algorithm [6]. Xing et al developed a novel mutation operator based on the immunity operation [7]. Deep and Das proposed a quadratic approximation based hybrid genetic algorithm for function optimization, in this paper they hybridized four GAs (GA1-GA4) by incorporating the quadratic approximation operator in to them resulting in four hybrid GAs, called HGA1-HGA4 which are compared with four simple GAs on a set of 22 numerical problems [8]. Deep et al proposed a new mutation operator for real coded genetic algorithms and its performance is compared with real coded Power Mutation operator [9]. Ahmed et al. established a hybrid HPSOM algorithm; the main idea of HPSOM is to integrate the PSO with genetic algorithm mutation method [10].

The proposed algorithm presents a hybrid algorithm based on exploration power of SOMA and exploitation feature of Non Uniform mutation. A set of 15 well known test problem has been used to evaluate the performance of M-SOMA and the comparison of this is done with SOMA and GA [3].

This paper is organized as follows. In section II, SOMA is described. In section III, the proposed algorithm M-SOMA is presented. In section IV, the numerical results are discussed. Finally, the paper concludes with section V drawing the conclusions of the present study.

II. SELF ORGANIZING MIGRATING ALGORITHM

Self Organizing migrating Algorithm is a population based stochastic search technique which is based on the social behavior of a group of individuals [11]. At each generation the individual with highest fitness value is known as leader and the worst is known as active. Rather than competing with each other, the active individual proceeds in the direction of the leader. This algorithm moves in migration loops and in each migration loop active individual travels a certain distance towards the leader in n steps of defined length. This path is perturbed randomly by PRT parameter. PRT vector is created before an individual proceeds towards leader. It is defined in the range (0,1). The movement of an individual is given as follows:

$$x_{i,j}^{MLnew} = x_{i,j,start}^{ML} + (\, x_{L,j}^{ML} - x_{i,j,start}^{ML}\,)\; t\, PRTVector_j \quad (1)$$

$Where\ \ t \in <0,\ by\ step\ to,\ PathLength>$

$ML\ is\ actual\ migration\ loop$

$x_{i,j}^{MLnew}$ $\ is\ the\ new\ positions\ of\ an\ individual.$

$x_{i,j,start}^{ML}$ $\ is\ the\ positions\ of\ active\ individual.$

$x_{L,j}^{ML}$ $\ is\ the\ positions\ of\ leader.$

The computational steps of SOMA are given as follows:
Step 1: generate initial population;
Step 2: evaluate all individuals in the population;
Step 3: generate PRT vector for all individuals;
Step 4: sort all of them;
Step 5: select the best fitness individual as leader and worst as active;
Step 6: for active individual new positions are created using "(1)". Then the best position is selected and replaces the active individual by the new one;
Step 7: if termination criterion is satisfied stop else go to step 2;
Step 8: report the best individual as the optimal solution;

III. PROPOSED M-SOMA ALGORITHM

In this section a hybridized SOMA, M-SOMA has been presented which uses Non Uniform mutation operator for creating the new solution member in the search space. As discussed in the introductory section that in the working of SOMA, no new solutions are created during the search instead only the positions of the solutions are changed. So, to avoid premature convergence and for maintaining the diversity of the population, new points using Non Uniform mutation operator are created in the search space.

A . Non Uniform Mutation Operator

This randomly selects one solution x_{ij} and sets its value according to the following rule:

$$x_{ij}^{0} = \begin{cases} x_{ij} + (ub_j - x_{ij}).\tau(t)\ if\ a_1 < 0.5 \\ x_{ij} - (x_{ij} + lb_j).\tau(t)\ if\ a_1 \geq 0.5 \end{cases} \quad (2)$$

$Where\ \ \tau(t) = \left(a_2\left(1\frac{t}{t_{max}}\right)\right)^{b}\ and\ x_{ij}\ \epsilon\left(lb_j, ub_j\right)$ with $a_1\ and\ a_2$ two random numbers in [0,1] , b a constant parameter, t the time or generation number, t_{max} the maximum number of generations any algorithm is allowed to run. The bounds (lower bound *lbj* and upper bound *ubj*) are taken equal to the bounds of the parameters of the problem to be optimized.

B. Hybridization

First the individuals are generated randomly. At each generation the individual with highest fitness value is selected as leader and the worst one as active individual. Now the active individual moves towards leader in n steps of defined length. The movement of this individual is given in "(1)". Then we again select the best and worst individual from the population. Now a new point is created using mutation using "(2)". This new point is accepted only if it is better than active individual and is replaced with active individual. In Fig. 1 flow chart of M-SOMA process is given. The computational steps of M-SOMA are given as follows:
Step 1: generate initial population;
Step 2: evaluate all individuals in the population;
Step 3: generate PRT vector for all individuals;
Step 4: sort all of them;
Step 5: select the best fitness individual as leader and worst as active;
Step 6: for active individual new positions are created using "(1)". Then the best position is selected and replaces the active individual by the new one;
Step 7: create new point by mutation using "(2)";
Step 8: if new point is better than active replace active with the new one;
Step 9: if termination criterion is satisfied stop else go to step2;
Step 10: report the best individual as the optimal solution;

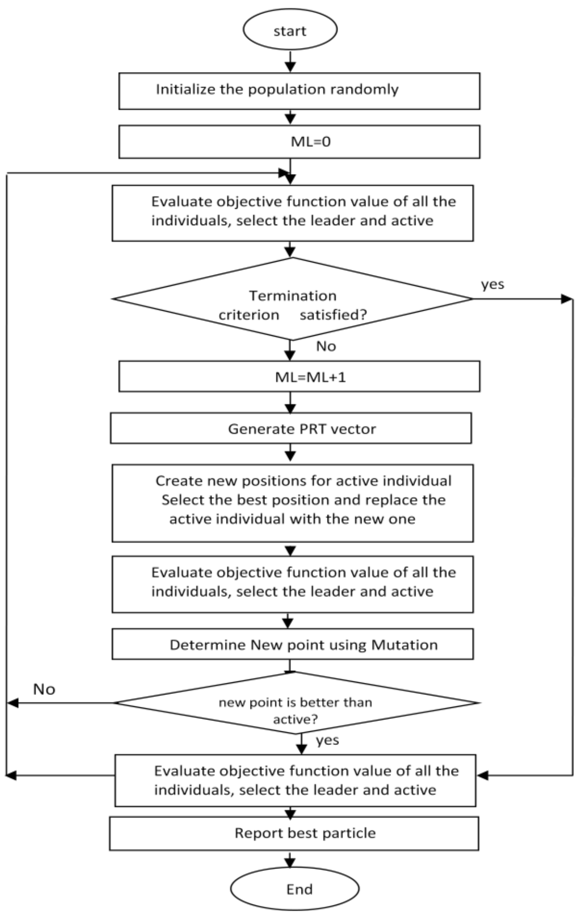
start
Initialize the population randomly
ML=0
Evaluate objective function value of all the individuals, select the leader and active
Termination criterion satisfied?
yes
No
ML=ML+1
Generate PRT vector
Create new positions for active individual Select the best position and replace the active individual with the new one
Evaluate objective function value of all the individuals, select the leader and active
Determine New point using Mutation
No
new point is better than active?
yes
Evaluate objective function value of all the individuals, select the leader and active
Report best particle
End

Fig. 1. Flowchart of M-SOMA Process

IV. NUMERICAL RESULTS ON BENCHMARK PROBLEMS

The proposed algorithm is coded in C++ and run on a PresarioV2000 1.50 GHz computer. The numerical results of the 15 problems are obtained. All the problems are of minimization and have the minimum value as 0. Since M-SOMA is probabilistic technique and rely heavily on the generation of random numbers, therefore 30 trials of each are carried out, each time using a different seed for the generation of random numbers. A run is considered to be a success if the optimum solution obtained falls within 1% accuracy of the known global optimal solution. The stopping criterion is either a run is a success or a fixed number of function calls (40,000) are performed.

The comparative performance of M-SOMA, Soma and GA are measured in terms of three criteria, namely accuracy, efficiency and reliability. They are described as follows:

1) Accuracy: which is based on average function values,
2) Efficiency: which is based on average number of function calls and
3) Reliability: which is based on the success rate of the algorithms?

Trials for the 15 problems are performed for dimension n=10. The value of parameters after fine tuning related to M-SOMA, namely population size, PRT, step size, path length and total number of function calls allowed for one run are shown in Table I. All the fifteen problems with their range of initialization are given in Table II.

Table III shows the number of successful runs of a total of 30 runs, corresponding to M-SOMA, GA and SOMA. Results show that the ranking of all the algorithms is GA < SOMA < M-SOMA. M-SOMA is best in 13 problems. Hence M-SOMA is *most reliable.*

Table IV shows the average number of function calls corresponding to M-SOMA, GA and SOMA. Results show that the ranking of all the algorithms is GA < SOMA < M-SOMA. M-SOMA is best in 7 problems. Hence M-SOMA is *most efficient.*

Table V shows the mean objective function value corresponding to M-SOMA, GA and SOMA. Results show that the ranking of all the algorithms is GA < SOMA < M-SOMA. M-SOMA is best in 14 problems. Hence M-SOMA is *most accurate*.

. The problems which could not be solved by the particular algorithm is given the symbol (*) at the corresponding entries. After analysis results are reported in Table VI. From the Table VI, It is very clear that M-SOMA outperforms GA and SOMA in all terms.

In order to reconfirm our results, we compare the relative performance of all the algorithms simultaneously. We use a Performance Index (PI). The relative performance of an algorithm using this modified PI is calculated in the following manner.

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}\left(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i\right) \qquad (3)$$

Where $\alpha_1^i = \frac{Sr^i}{Tr^i}$,

$$\alpha_2^i = \begin{cases}\frac{Mo^i}{Ao^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0\end{cases} \quad and$$

$$\alpha_3^i = \begin{cases}\frac{Mt^i}{At^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0\end{cases} \quad and$$

Where

Sr^i = Number of successful runs of *ith* problem

Tr^i = Total number of runs of *i*th problem

Ao^i = Mean objective function value obtained by an algorithm of *i*th problem

Mo^i = Minimum of Mean objective function value obtained by all algorithms of *i*th problem

At^i= Mean execution time of successful runs taken by an algorithm in obtaining the solution of *i*th problem

Mt^i= Minimum of mean execution time of successful runs taken by all algorithms in obtaining the solution of *i*th problem

N_p = Total number of problems analyzed

Table I Parameters of M-SOMA

Dimension	10
Population Size	10
PRT	0.1, 0.3, 0.5, 0.9
Step	0.11 and 0.31
Path length	3
Total number of function calls allowed	40,000
b	0.6 to 5

Table II Benchmark Functions

S.No.	Name	Function	Range
1	Ackley	$-20\ exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - exp(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi x_i)) + 20 + e$	[-30, 30]
2	Cosine Mixture	$0.1n + \sum_{i=1}^{n}x_i^2$ - $0.1\sum_{i=1}^{n}\cos(5\pi x_i)$	[-1, 1]

3	Exponential	$1-(\exp(-0.5\sum_{i=1}^{n}x_i^2))$	[-1, 1]
4	Griewank	$\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]
5	Paviani	$45.778+\sum_{i=1}^{10}[(\ln(x_i-2))^2+(\ln(10-x_i)^2)]-(\prod_{i=1}^{n}x_i)^{0.2}$	[2, 10]
6	Rastrigin	$10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	[-5.12, 5.12]
7	Rosenbrock	$\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	[-30, 30]
8	Schwefel	$418.9829-\sum_{i=1}^{n}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500, 500]
9	Sinusoidal	$3.5-\left[2.5\prod_{i=1}^{n}sin\left(x_i-\frac{\pi}{6}\right)+\prod_{i=1}^{n}sin\left(5\left(x_i-\frac{\pi}{6}\right)\right)\right]$	[0, π]
10	Sphere Function	$\sum_{i=1}^{n}x_i^2$	[-5.12, 5.12]
11	Axis parallel hyper ellipsoid	$\sum_{i=1}^{n}ix_i^2$	[-5.12, 5.12]
12	Schwefel's double sum	$\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	[100, 100]
13	De-jong's function with noise	$\sum_{i=1}^{n}\left((1+i)x_i^4+rand(0,1)\right)$	[-10, 10]
14	Ellipsoidal	$\sum_{i=1}^{n}(x_i-i)^2$	[-n, n]
15	Dixon and Price function	$(x_1-1)^2+\sum_{i=2}^{n}i(2x_i^2-x_{i+1})^2$	[-10, 10]

Table III Percentage of success of M-SOMA, GA and SOMA

Problem Number	No. of successful runs out of 30			Best amongst GA SOMA & M-SOMA
	GA	SOMA	M-SOMA	
1	11	11	30	M-SOMA
2	5	25	30	M-SOMA
3	30	30	30	ALL
4	0	18	19	M-SOMA
5	0	30	30	SOMA & M-SOMA
6	0	9	20	M-SOMA
7	0	1	5	M-SOMA
8	0	6	30	M-SOMA
9	0	0	30	M-SOMA
10	30	30	30	ALL
11	30	30	30	ALL
12	30	30	16	GA & SOMA
13	5	0	26	M-SOMA
14	0	30	30	SOMA & M-SOMA
15	0	25	23	SOMA

Table IV Average number of function evaluations of GA, SOMA and M-SOMA

Problem Number	Average no. of function evaluations of successful runs			Best amongst GA SOMA & M-SOMA
	GA	SOMA	M-SOMA	
1	9539	15297	22126	GA
2	4807	5968	9780	GA
3	2400	4206	3830	GA
4	*	33706	7614	M-SOMA
5	*	9681	4464	M-SOMA
6	*	12997	33601	SOMA
7	*	48853	18291	M-SOMA
8	*	13213	15429	SOMA

9	*	*	5159	M-SOMA
10	4394	7486	4231	M-SOMA
11	4229	6085	5385	GA
12	5712	12857	19698	GA
13	13510	*	23475	GA
14	*	9155	6040	M-SOMA
15	*	36995	22901	M-SOMA

Table V Mean and Standard Deviation of objective function value of M-SOMA, GA and SOMA

Problem	GA	SOMA	M-SOMA	GA	SOMA	M-SOMA	Algorithm with
Number	Mean	S.D	Mean	S.D.	Mean	S.D.	least mean
1	1.243	0.980	0.000709	0.990	0.007	0.000158	M-SOMA
2	0.287	0.246	0.000568	0.082	0.004	0.000271	M-SOMA
3	0.005	0.004	0.000794	0.002	0.006	0.000183	M-SOMA
4	0.365	0.315	0.00763	0.008	0.068	0.00192	M-SOMA
5	6.465	4.160	0.000686	0.003	0.005	0.000265	M-SOMA
6	21.319	7.721	0.00229	1.139	0.003	0.00264	M-SOMA
7	54.340	53.071	0.0959	2.478	4.699	0.0030	M-SOMA
8	690.353	210.977	0.00070	136.584	0.004	0.000201	M-SOMA
9	0.915	0.490	0.000753	0.116	0.193	0.000266	M-SOMA
10	0.002	0.003	0.000800	0.003	0.007	0.000216	M-SOMA
11	0.002	0.003	0.000808	0.026	0.006	0.000201	M-SOMA
12	0.001	0.002	0.00962	0.003	0.007	0.00032	SOMA
13	0.269	0.255	0.00863	0.069	0.162	0.000197	M-SOMA
14	7.683	5.766	0.000827	0.002	0.006	0.000157	M-SOMA
15	28.568	40.029	0.00921	0.278	0.134	0.00071	M-SOMA

Table VI Comparison of M-SOMA, GA and SOMA

Factors	Performance of M-SOMA	M-SOMA Vs GA	M-SOMA Vs SOMA	Overall performance of M-SOMA, GA &SOMA
Success Rate	Better	11	08	GA: 04
(Ref. Table III)	Equal	03	05	SOMA: 07
	Worse	01	02	M-SOMA: 13
Average number of	Better	09	11	GA: 06
function calls	Equal	00	00	SOMA: 02
(Ref. Table IV)	Worse	06	04	M-SOMA: 07
Mean function value	Better	14	14	GA: 00
(Ref. Table VI)	Equal	00	00	SOMA: 01
	Worse	01	01	M-SOMA: 14

$k_{1,}k_2$ *and* k_3 ($k_1 + k_2 + k_3 = 1$ *and* $0 \le k_1, k_2, k_3 \le 1$) are the weights assigned to percentage of success, mean objective function value and mean execution time of successful runs, respectively

From the above definition it is clear that modified PI is a function of $k_{1,}k_2$ *and* k_3 *since* $k_1 + k_2 + k_3 = 1$, *one of* k_i, $i = 1,2,3$ could be eliminated to reduce the number of variables from the expression of PI. But it is still difficult to analyze the behavior of this PI , because the surface of PI for all the algorithms are overlapping and it is difficult to visualize them. Hence equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows:

(i) $$k_1 = w,\ k_2 = k_3 = \frac{1-w}{2}, 0 \le w \le 1$$

(ii) $$k_2 = w,\ k_1 = k_3 = \frac{1-w}{2}, 0 \le w \le 1$$

(iii) $$k_3 = w,\ k_1 = k_2 = \frac{1-w}{2}, 0 \le w \le 1$$

The graphs corresponding to each of case (i), (ii) and (iii) are shown in Figs. 2.1, 2.2 and 2.3. The horizontal axis represents the weight w and the vertical axis represents the performance index PI.

In case (i), the mean objective function value and mean execution time of successful runs are given equal weights. PI's of M-SOMA, GA and SOMA are superimposed in the Fig. 2.1. It is observed that the value of PI for M-SOMA is more than GA and SOMA.

In case (ii), equal weights are assigned to the numbers of successful runs and mean execution time of successful runs. From Fig.2.2 it is clear that M-SOMA has the highest PI.

In case (iii), equal weights are assigned to mean objective function value and average number of successful runs. From Fig. 2.3 it is clear that M-SOMA has the highest PI.

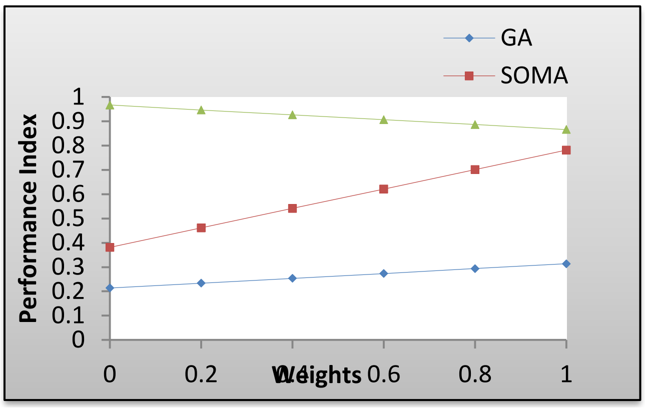

Fig. 2.1 PI for combination of M-SOMA, GA and SOMA for case1

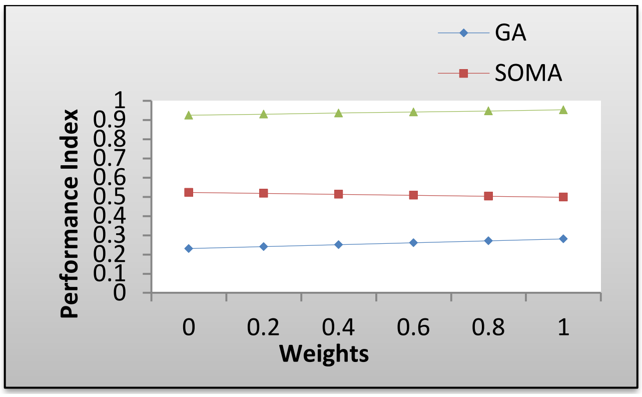

Fig. 2.2 PI for combination of M-SOMA, GA and SOMA for case 2

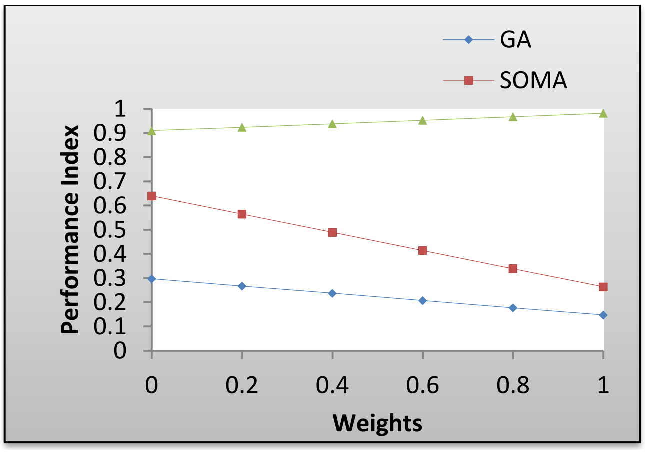

Fig. 2.3 PI for combination of M-SOMA, GA and SOMA for case 3

V. CONCLUSION

In this paper hybridization of SOMA and Non Uniform mutation is proposed. The proposed algorithm is tested on 15 unconstrained benchmark problems and compared with the results of GA and SOMA. The results are obtained by using population size 10 only. On the basis of the results it can be concluded that the presented algorithm outperforms GA and SOMA efficiency, reliability and accuracy.

REFERENCES

1. Zelinka,, J. Lampinen," SOMA- Self Organizing Migrating Algorithm", in proceedings of the 6th International Mendel Conference on Soft Computing, 2000, pp. 177-187, Brno, Czech, Republic .
2. G. P. Kasprzyk and M. Jasku, "Application of Hybrid Genetic Algorithms for Deconvulation of Electrochemical Responses in SSLSV Method", vol. 567, Journal of Electroanalytical chemistry, 2004, pp. 39 – 66.
3. K. Deep and Dipti, "A New Hybrid Self Organizing Migrating Genetic Algorithm for Function Optimization", IEEE Congress on Evolutionary Computation, 2007, pp. 2796-2803.

A. *Khosravi, A. Lari and J. Addeh , " A New Hybrid of Evolutionary and Conventional Optimization Algorithm", vol. 6, Applied Mathematical Sciences, 2012, pp. 815-825 .*

4. S. Ghatei, et. al., "A New Hybrid Algorithm for Optimization using PSO and GDA", Journal of Basic and vol. 2, Applied Scientific Research, 2012, pp. 2336-2341.
5. K. Deep and M. Thakur, "A new mutation operator for real coded genetic algorithms", vol. 193, Applied Mathematics and computation, 2007, pp. 229-247.
6. L. N. Xing, Y. W. Chen and K.W. Yang, "A novel mutation operator base on immunity operation", vol. 197, European Journal of Operational Research, 2009, pp. 830-833.
7. K. Deep and Kedar Nath Das, " Quadratic approximation based Hybrid Genetic Algorithm Function Optimization", vol. 203, Applied Mathematics and Computations, 2008 , pp. 86-98, Elsevier.
8. K. Deep, Shashi and V. K. Katiyar, "A new real coded genetic algorithm operator: Log logistic mutation", In proceedings of the international conference on soft computing for problem solving, vol. 130, Advances in intelligent and Soft Computing, 2012 , pp. 193-200.

A. A. Ahmed Esmin and Stan Matwin ,"A Hybrid Particle Swarm Optimization Algorithm with Genetic Mutation", International Journal of Innovative Computing, Information and Control, vol. 9, 2013, pp. 1919-1934.

9. Zelinka, "SOMA- Self Organizing Migrating Algorithm", in New optimization techniques in engineering, G. C. Onwubolu and B.V. Babu, Eds. Berlin, Germany: Springer (2004).

AUTHORS PROFILE

Dipti Singh, received the M.Sc degree in Applied Mathematics from Indian Institute of Technology, Roorkee in 2002 and the Ph.D. degree in Optimization Techniques from Indian Institute of Technology, Roorkee in 2007. She was one of the faculties of Amity University, Noida and is currently an Assistant professor at Gautam Buddha University, Greater Noida. Her research interests include optimization techniques, computational intelligence, and their application to real life engineering problems.

Seema Agrawal, received the M.Sc degree in Applied Mathematics and M.Phil. degree in computer applications from Indian Institute of Technology, Roorkee. She was one of the faculties of Sharda University, Greater Noida and is currently an Assistant professor at S.S.V.(P.G.) College, Hapur. Her research interests include optimization techniques, computational intelligence, and their application to real life engineering problems.