Self Organizing Migrating Algorithm with Nelder Mead Crossover and Log-Logistic Mutation for Large Scale Optimization

Dipti Singh and Seema Agrawal

Abstract. This chapter presents a hybrid variant of self organizing migrating algorithm (NMSOMA-M) for large scale function optimization, which combines the features of Nelder Mead (NM) crossover operator and log-logistic mutation operator. Self organizing migrating algorithm (SOMA) is a population based stochastic search algorithm which is based on the social behavior of group of individuals. The main characteristics of SOMA are that it works with small population size and no new solutions are generated during the search, only the positions of the solutions are changed. Though it has good exploration and exploitation qualities but as the dimension of the problem increases it trap to local optimal solution and may suffer from premature convergence due to lack of diversity mechanism. This chapter combines NM crossover operator and log-logistic mutation operator with SOMA in order to maintain the diversity of population and to avoid the premature convergence. The proposed algorithm has been tested on a set of 15 large scale unconstrained test problems with problem size taken as up to 1000. In order to see its efficiency over other population based algorithms, the results are compared with SOMA and particle swarm optimization algorithm (PSO). The comparative analysis shows the efficiancy of the proposed algorithm to solve large scale function optimization with less function evaluations.

1 Introduction

Self Organizing Migrating Algorithm (SOMA) is a stochastic population based algorithm based on the social behavior of a group of individuals presented by Zelinka

Dipti Singh
Department of Applied Sciences, Gautam Buddha University, Greater Noida, India
e-mail: diptipma@rediffmail.com

Seema Agrawal
Department of Mathematics, S.S.V.P.G. College, Hapur, India
e-mail: Seemagrwl7@gmail.com

D.P. Acharjya et al. (eds.), *Computational Intelligence for Big Data Analysis*,
Adaptation, Learning, and Optimization 19, DOI: 10.1007/978-3-319-16598-1_6

and Lampinen in 2000 [1]. This algorithm is inspired by the competitive cooperative behavior of intelligent creatures solving a common problem. Such a behavior can be observed anywhere in the world. 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 (it has found some food or shelter) 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. 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. SOMA converges very fast for small scale problems but as the problem size increases its convergence becomes very slow and it may trap to local optima. It is because of poor balancing between exploration and exploitation. Exploration and exploitation are considered as two major characteristics of population based algorithms for maintaining the diversity of the population and to speed up the convergence.

To overcome the difficulty of premature convergence and to avoid the loss of diversity of population in search space one simple way is to combine population based algorithms with the features of other population based algorithms or hybridized them with local search algorithms. In this regard, many attempts have been made in past. Domingo proposed a real coded crossover operator for evolutionary algorithms based on the statistical theory of population distributions [2]. Chelouah and Siarry proposed a hybrid method that combines the feature of continuous Tabu search and Nelder-Mead Simplex algorithm for the global optimization of multiminima functions [3]. Deep and Dipti proposed a hybridized variant SOMGA for function optimization which combines the features of binary coded GA and real coded SOMA [4]. Fan et al. hybridized Nelder-Mead Simplex method with Genetic algorithm and particle swarm optimization to locate the global optima of non-linear continuous variable functions [5]. Premalatha and Nataranjan established a hybrid variant of PSO that proposes the modification strategies in PSO using GA to solve the optimization problems [6]. Khosravi et al. proposed a novel hybrid algorithm by combining the abilities of evolutionary and conventional algorithm simultaneously [7]. Ghatei et al. designed a new hybrid algorithm using PSO and GDA, in which global search character of PSO and local search factor of Great Deluge Algorithm are used based on series [8] . Ahmed et al. proposed a hybrid HPSOM algorithm, in which PSO is integrated with genetic algorithm mutation method [9]. Millie et al. presented a variant of quantum behaved particle swarm optimization (Q-QPSO) for solving global optimization problems which is based on the characteristics of QPSO, and uses interpolation based recombination operator for generating a new solution vector in the search space [10]. Deep and Bansal developed a variant of PSO with hybridization of quadratic approximation operator for economic dispatch problems with valve-point effects [11]. Deep and Thakur proposed a new mutation operator for real coded genetic algorithm [12]. Xing et al. developed a novel mutation operator based on the immunity operation [13]. Deep et al. proposed a new mutation operator for real coded genetic algorithms and its performance is compared with real coded power mutation operator [14]. Mohan and Shankar developed random search technique for global optimization based on quadratic approximation [15]. 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 [16]. Deep and Bansal presented the hybridization of PSO with quadratic approximation operator (QA), the hybridization is performed by splitting the whole swarm into two subswarms [17]. To improve the performance of real coded genetic algorithm Deep and Das hybridized it with quadratic approximation [18]. Millie et al. presented a new variant of particle swarm optimization named QPSO for solving global optimization problems [19]. There has not been done much work on hybridization of SOMA with other approaches except [4]. Recently Singh Dipti et al. presented a variant (SOMAQI) of SOMA, in which SOMA is combined with quadratic interpolation crossover in order to improve its efficiency for finding the solution of global optimization problems of small scale [20].

In this chapter, again a variant NMSOMA-M of SOMA has been proposed, in which SOMA is hybridized with NM crossover operator and Log-logistic mutation operator. The proposed algorithm, not only remove the difficulty of premature convergence of large scale function optimization but also maintain the diversity of the population. In this approach a new linear NM crossover operator has been designed to create a new member in the search space and has been used along with Log-logistic mutation operator to maintain the diversity of the populations during the search. Its efficiency has been tested on 15 scalable unconstrained test problems with problem size vary up to 1000 and a comparative analysis has been made between PSO, SOMA and proposed algorithm. The information about PSO is given in [21].

The chapter is organized as follows: In section 2, SOMA is described. In section 3, the methodology of proposed Algorithm NMSOMA-M has been discussed in detail. Test problems used for testing of the proposed algorithm has been listed in section 4. In section 5, numerical results of the present study have been discussed. Finally, the chapter concludes with Section 6 drawing the conclusions of the present study.

2 Self Organizing Migrating Algorithm

Self organizing migrating algorithm is relatively a new stochastic evolutionary algorithm which is based on the social behavior of a group of individuals [22]. Inspired by the competitive cooperative behavior of intelligent creatures, the working of this algorithm is very simple. At each generation the individual with highest fitness value is known as leader and the worst is known as active is taken into consieration. 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 a parameter known as *PRT* parameter. It is defined in the range of $< 0,1 >$. A PRT vector is created using this PRT parameter value before an individual proceeds towards leader, known as perturbation vector. This randomly generated binary perturbation vector controls the allowed dimensions for an individual of population. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension. To create this perturbation vector, following expression is used:

```
if $rnd_j < PRT$ then
        $PRTVector_j = 1$
else
        $PRTVector_j = 0$
```

The movement of an individual during the migration is given as follows:

$$x_{i,j}^{MLnew} = x_{i,jstart}^{ML} + (x_{i,j}^{ML} - x_{i,jstart}^{ML})tPRTVector_j \tag{1}$$

Where $t \in < 0, bystepto, pathlength >$, ML is actual migration loop, $x_{i,j}^{MLnew}$ is the new positions of an individual, $x_{i,jstart}^{ML}$ is the positions of active individual and $x_{i,j}^{ML}$ is the positions of leader. The computational steps of SOMA are given as follows. The flow chart of SOMA process is depicted in Figure 1:

Algorithm (SOMA)

1. Generate initial population
2. Evaluate all individuals in the population
3. Generate PRT vector for all individuals
4. Sort all of them
5. Select the best fitness individual as leader and worst as active
6. For active individual new positions are created using equation (1). Then the best position is selected and replaces the active individual by the new one if it is better than active individual
7. If termination criterion is satisfied stop else go to Step-2
8. Report the best individual as the optimal solution

3 Proposed NMSOMA-M Algorithm

In this section a new hybrid variant of SOMA, NMSOMA-M has been presented which uses NM crossover operator and log logistic mutation operator for creating the new solution member in the search space. As discussed earlier, in the working of SOMA, no new solutions are created during the search instead only the positions of the solutions are changed. Due to which there is loss of diversity in the population as we move on to solve large scale optimization problem. So, to avoid premature convergence and for maintaining the diversity of the population, new points are created in the search space using NM crossover operator and log logistic mutation operator.

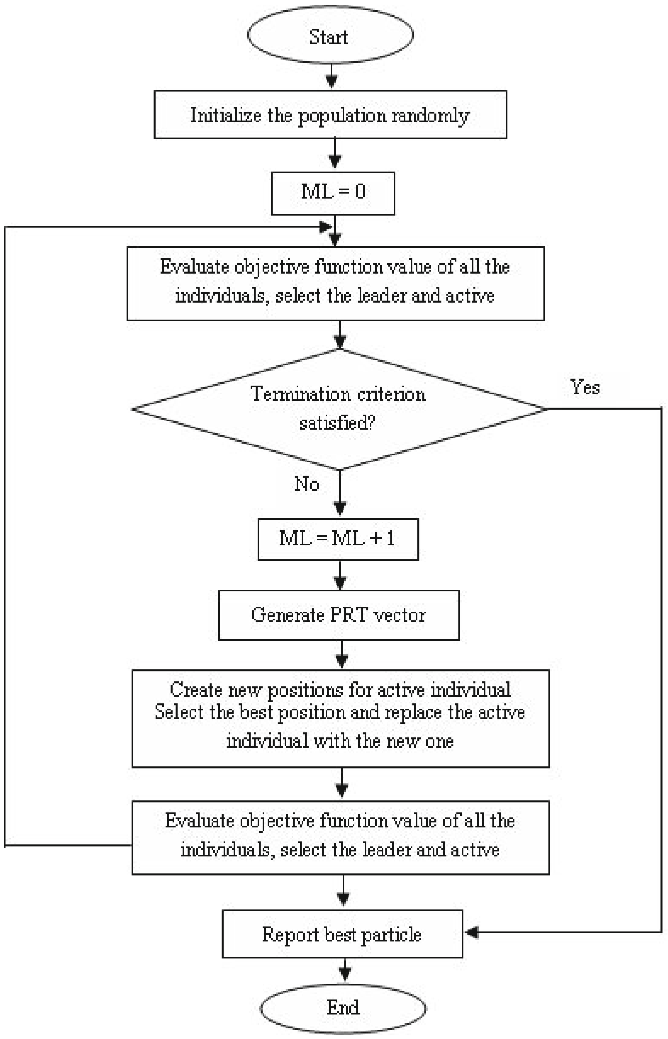
Start
Initialize the population randomly
ML = 0
Evaluate objective function value of all the individuals, select the leader and active
Termination criterion satisfied?
Yes
No
ML = ML + 1
Generate PRT vector
Create new positions for active individual Select the best position and replace the active individual with the new one
Evaluate objective function value of all the individuals, select the leader and active
Report best particle
End

Fig. 1 Flow chart of SOMA process

3.1 Nelder Mead (NM) Crossover Operator

The Nelder Mead simplex (NM) method is a computational algorithm and is based upon the work of Spendley et al. [23]. It forms a simplex and uses this simplex to search for a local minimum. A simplex is defined as a geometrical figure which is formed by $(N+1)$ vertices, where N is the number of variables of a function. Through a sequence of elementary geometric transformation (reflection, contraction, expansion), the initial simplex moves, expands or contracts. After each transformation, the current worst vertex is replaced by a better one. In the proposed work, Nelder Mead simplex search method has been used as a linear NM crossover operator which uses two out of three randomly chosen members from population of individuals to create a new member. The computational steps of NM crossover operator method are as follows:

Algorithm (NM Crossover Operator)

```
1. Choose parameters $\alpha$, $\beta$ and $\gamma$
2. Select an initial simplex with randomly chosen three vertices
3. Calculate function values at chosen vertices
4. Find $x_h$ (the worst point), $x_l$ (the best point), $x_g$ (next to the worst point) and
   evaluate the value of objective functions $f_h$, $f_l$ and $f_g$ at these points
5. Compute $x_c$ which is the centroid of $x_l$ and $x_g$
6. The NM uses three operators; reflection, contraction and expansion to improve
   the worst point. Compute the reection point $x_r$ by using the following expression
   and then compute the value of objective function $f_r$.
         $x_r = x_c + \alpha(x_c - x_h)$        (Reflection)
   if $f_r < f_l$
             $x_{new} = (1+\gamma)x_c - \gamma x_h$        (Expansion)
   else if $f_r \geq f_h$
             $x_{new} = (1-\beta)x_c + \beta x_h$        $(Outside\ contraction)$        (2)
     else if $f_g < f_r < f_h$
             $x_{new} = (1+\beta)x_c - \beta x_h$        (Inside contraction)
   Calculate $f_{new}$ and replace $x_h$ by $x_{new}$ if $f_{new} < f_h$
7. The method continues until reaching some stopping criteria
```

3.2 Log Logistic Mutation Operator

The mutation operator [14] has been taken into consideration and adopted in this chapter. This randomly selects one solution x_{ij} and sets its value according to the following rule:

$$x_{ij}^{new} = \begin{cases} x_{ij} + \lambda(u - x_{ij}) & if \quad r \geq T \\ x_{ij} - \lambda(x_{ij} - l) & if \quad r < T \end{cases} \tag{3}$$

where $r \in (0,1)$ is uniformly distributed random number, u and l are the upper and lower bounds of the decision variable, $T = (x_{ij} - l)/(u - x_{ij})$ and λ is a random number following log logistic distribution and is given as equation 4.

$$\lambda = b\left(\frac{h}{1-h}\right)^{\frac{1}{\alpha}} \tag{4}$$

Where $h \in (0,1)$ is a uniformly distributed random number, $b > 0$ is a scale parameter and α is termed as mutation index as it controls the strength of mutation. More information on this operator can be found in [14].

3.3 Methodology of the Proposed Algorithm NMSOMA-M

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 equation (1). Again the best and worst individual from the population is selected. Now a new point is created using Nelder Mead crossover operator using equation (2). This new point is accepted only if it is better than active individual and is replaced with active individual. Then again the best and worst individual from the population is selected. Now a new point is created using log logistic mutation operator using equation (3). This new point is accepted only if it is better than active individual and is replaced with active individual. The computational steps of NMSOMA-M are given below. The flowchart of the proposed algorithm NMSOMA-M process is depicted in Figure 2.

Algorithm(NMSOMA-M)

1. Generate initial population
2. Evaluate all individuals in the population
3. Generate PRT vector for all individuals
4. Sort all of them
5. Select the best fitness individual as leader and worst as active
6. For active individual new positions are created by using equation (1). The best position is selected and replaces the active individual by the new one
7. Create new point by crossover operator as defined in equation (2)
8. If new point is better than active, replace active with the new one
9. Create new point by mutation operator as defined in equation (3)
10. If new point is better than active, replace active with the new one
11. If termination criterion is satisfied then terminate the process; else go to step 2
12. Report the best individual as the optimal solution

4 Benchmark Functions

In this section the set of 15 scalable benchmark functions have been listed. These problems vary in nature and their complexity. The performance of proposed algorithm has been evaluated on the following functions which can be formulated as follow:

1. Ackley function

$$min f(x) = -20\exp\left(-0.02\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

for $x_i \in [-30,30], x^* = (0,0,\cdots,0), f(x^*) = 0$

2. Cosine Mixture

$$minf(x) = 0.1n + \sum_{i=1}^{n} x_i^2 - 0.1\sum_{i=1}^{n}\cos(5\pi x_i)$$

for $x_i \in [-1,1], x^* = (0,0,\cdots,0), f(x^*) = 0$

3. Exponential

$$minf(x) = 1 - \exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$$

for $x_i \in [-1,1], x^* = (0,0,\cdots,0), f(x^*) = 0$

4. Griewank

$$minf(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$

for $x_i \in [-600,600], x^* = (0,0,\cdots,0), f(x^*) = 0$

5. Levy and Montalvo 1

$$minf(x) = \frac{\pi}{n}\left(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_n-1)^2\right);$$

$y_i = 1 + \frac{1}{4}(x_i+1)$
for $x_i \in [-10,10], x^* = (0,0,\cdots,0), f(x^*) = 0$

6. Levy and Montalvo 2

$$minf(x) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]\right.$$
$$\left. +(x_n-1)^2[1+\sin^2(2\pi x_n)]\right)$$

for $x_i \in [-5,5], x^* = (0,0,\cdots,0), f(x^*) = 0$

7. Rastrigin

$$minf(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)]$$

for $x_i \in [-5.12,5.12], x^* = (0,0,\cdots,0), f(x^*) = 0$

8. Rosenbrock

$$minf(x) = \sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2 + (x_i-1)^2]$$

for $x_i \in [-30,30], x^* = (0,0,\cdots,0), f(x^*) = 0$

9. Schwefel 3

$$minf(x) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|$$

for $x_i \in [-10,10], x^* = (0,0,\cdots,0), f(x^*) = 0$

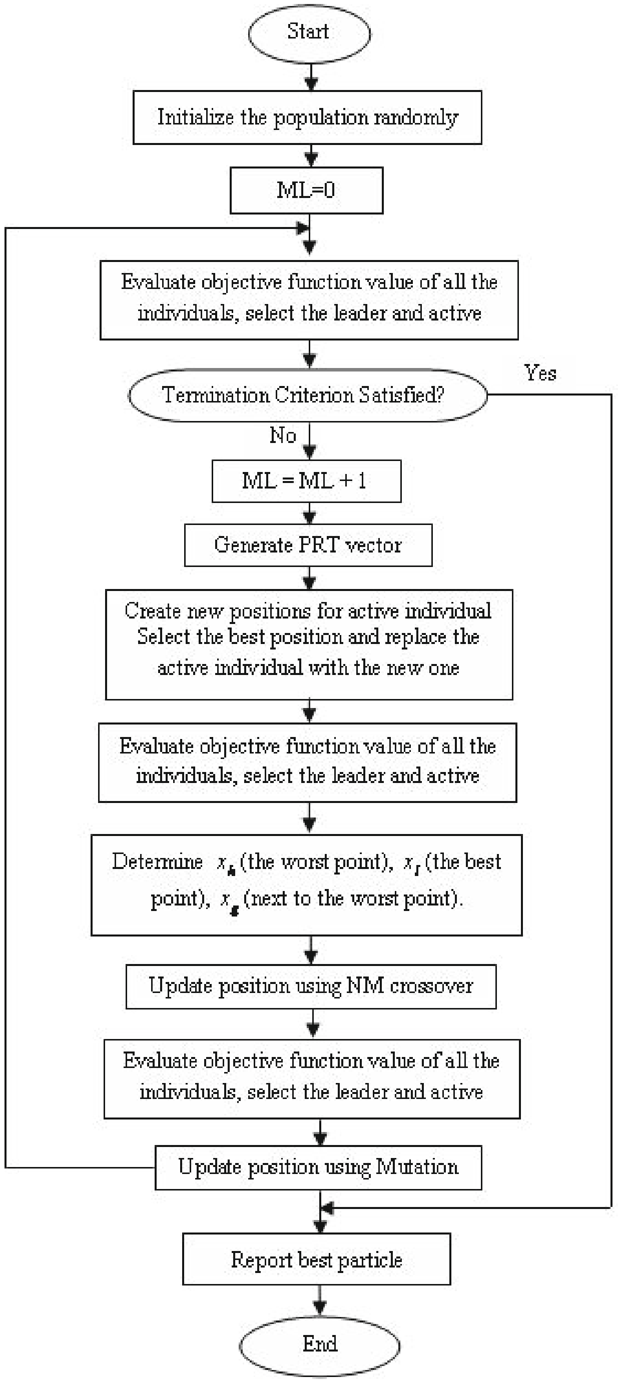
Start
Initialize the population randomly
ML=0
Evaluate objective function value of all the individuals, select the leader and active
Termination Criterion Satisfied?
Yes
No
ML = ML + 1
Generate PRT vector
Create new positions for active individual Select the best position and replace the active individual with the new one
Evaluate objective function value of all the individuals, select the leader and active
Determine x_h (the worst point), x_l (the best point), x_g (next to the worst point).
Update position using NM crossover
Evaluate objective function value of all the individuals, select the leader and active
Update position using Mutation
Report best particle
End

Fig. 2 Flowchart of NMSOMA-M process

10. De-Jongs function with noise

$$minf(x) = \sum_{i=0}^{n-1} (i+1)x_i^4 + rand(0,1)$$

for $x_i \in [-1.28, 1.28], x^* = (0,0,\cdots,0), f(x^*) = 0$

11. Step function

$$minf(x) = \sum_{i=1}^{n} \left(x_i + \frac{1}{2}\right)^2$$

for $x_i \in [-100, 100], x^* = (0.5, 0.5, \cdots, 0.5), f(x^*) = 0$

12. Sphere function

$$minf(x) = \sum_{i=1}^{n} x_i^2$$

for $x_i \in [-5.12, 5.12], x^* = (0,0,\cdots,0), f(x^*) = 0$

13. Axis parallel hyper ellipsoid

$$minf(x) = \sum_{i=1}^{n} ix_i^2$$

for $x_i \in [-5.12, 5.12], x^* = (0,0,\cdots,0), f(x^*) = 0$

14. Ellipsoidal

$$minf(x) = \sum_{i=1}^{n} (x_i - i)^2$$

for $x_i \in [-n, n], x^* = (1,2,\cdots,n), f(x^*) = 0$

15. Brown

$$minf(x) = \sum_{i=1}^{n-1} \left[\left(x_i^2\right)^{(x_{i+1}^2+1)} + \left(x_{i+1}^2\right)^{(x_i^2+1)} \right]$$

for $x_i \in [-1, 4], x^* = (0,0,\cdots,0), f(x^*) = 0$

5 Numerical Results on Benchmark Problems

In this section NMSOMA-M has been used to solve 15 benchmark problems in order to estimate its efficiency. For this purpose the dimension of the problems varies from 30 to 1000. In almost all problems except Griewank function the complexity of the problem increases as the dimension of the problem increases. So on the basis of the level of complexity, problems with dimension 30 is considered as small scale and 1000 as large scale. The proposed algorithm is coded in $C++$ and run on a Presario V2000 1.50 GHz computer. Since NMSOMA-M is probabilistic technique and relies 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 of success or a fixed number of migrations $10,000$. Maximum number of function evaluations allowed are taken as $1,50,000$. For fair comparison all parameters are kept same for all three algorithms. The comparative performance of these three algorithms is measured in terms of three criteria, namely accuracy, efficiency and reliability. They are described as follows:

1. Accuracy: It is based on average of mean objective function values of the successful runs
2. Efficiency: It is based on average number of function evaluations and
3. Reliability: This is based on the success rate of the algorithms

In the beginning, trials for the 15 problems are performed for dimension $n =$ 30, 50 and 100. The value of parameters after fine tuning related to NMSOMA-M, namely population size, PRT, step size, path length and total number of migrations allowed for one run are shown in Table 1.

Table 1 Parameters of NMSOMA-M

Parameters	Values
Dimension	30, 50, 100
Population size	10
PRT	0.1, 0.3, 1
Step, Path length	0.31, 3
Total number of migrations allowed	10,000
α, β and γ	1.8, 0.8 and 1

The number of successful runs (out of a total of 30 runs) taken by NMSOMA-M, PSO and SOMA for dimensions 30, 50 and 100 has been presented in Tables 2, 3 and 4 respectively. From table 2 it is clear that the performance of NMSOMA-M is better than SOMA and PSO. PSO shows worst performance out of three. The main reason behind the failure of PSO in many problems can be considered as population size. PSO requires large population size to work in comparison with SOMA. But, in the proposed method for solving 30, 50, 100, and 1000 dimension problem, only 10 population size is required. Similar kind of behavior can also be seen in table 5 and 8 respectively. The performance of SOMA and PSO start deteriorating as the complexity of the problem increases with rise in problem size. PSO almost fail to solve problems with dimension 100. On the basis of this analysis, these three algorithms can be ranked as PSO $<$ SOMA $<$ NMSOMA-M. Hence NMSOMA-M is most reliable.

In Tables 5, 6 and 7, the average number of function evaluations taken by NMSOMA-M, PSO and SOMA for dimensions 30, 50 and 100 respectively has been presented. From these three tables it is clear that NMSOMA-M attained desirable success in much lesser function evaluations as compared to SOMA and PSO. Since NMSOMA-M works with small population size, the function evaluations required is also very less. The algorithms can be ranked as PSO $<$ SOMA $<$ NMSOMA-M. Hence NMSOMA-M is most efficient.

Table 2 Percentage of success for dimension 30

Problem number	Number of successful runs out of 30		
	PSO	SOMA	NMSOMA-M
1	11	24	**30**
2	16	29	**30**
3	30	30	**30**
4	12	09	**30**
5	05	30	**30**
6	23	**29**	19
7	0	0	**30**
8	0	0	0
9	01	30	**30**
10	01	23	**30**
11	17	29	**30**
12	30	30	**30**
13	22	30	**30**
14	0	30	**30**
15	22	30	**30**

Table 3 Percentage of success for dimension 50

Problem number	Number of successful runs out of 50		
	PSO	SOMA	NMSOMA-M
1	0	06	**30**
2	0	23	**30**
3	30	30	**30**
4	03	08	**30**
5	0	30	**30**
6	16	**24**	09
7	0	0	**29**
8	0	0	0
9	0	30	**30**
10	0	0	**30**
11	06	30	**30**
12	30	30	**30**
13	18	30	**30**
14	0	30	**30**
15	11	30	**30**

Table 4 Percentage of success for dimension 100

Problem number	Number of successful runs out of 100		
	PSO	SOMA	NMSOMA-M
1	0	0	**30**
2	0	0	**30**
3	0	30	**30**
4	0	06	**30**
5	0	23	**30**
6	0	**07**	03
7	0	0	**28**
8	0	0	0
9	0	0	**30**
10	0	0	**30**
11	0	22	**30**
12	14	30	**30**
13	0	30	**30**
14	0	28	**30**
15	0	30	**30**

Table 5 Average number of function evaluations for dimension 30

Problem number	Average number of function evaluations		
	PSO	SOMA	NMSOMA-M
1	106423	32915	**422**
2	105392	15594	**339**
3	97650	6993	**382**
4	107052	22829	**595**
5	82958	10530	**5543**
6	92192	**13075**	20479
7	150000	150000	**1202**
8	150000	150000	150000
9	96440	21623	**700**
10	131170	94878	**5417**
11	107075	17341	**683**
12	93328	14159	**341**
13	96291	15695	**540**
14	150000	**16125**	37049
15	96395	15612	**848**

Table 6 Average number of function evaluations for dimension 50

Problem number	Average number of function evaluations		
	PSO	SOMA	NMSOMA-M
1	150000	50638	**603**
2	150000	23735	**645**
3	150000	12823	**445**
4	126573	42164	**568**
5	150000	18808	**9724**
6	111745	**20708**	73269
7	150000	150000	**1304**
8	150000	150000	150000
9	150000	39881	**699**
10	150000	150000	**2584**
11	123406	45490	**606**
12	116734	27072	**372**
13	117857	33949	**514**
14	150000	**35493**	60161
15	120549	24748	**758**

Table 7 Average number of function evaluations for dimension 100

Problem number	Average number of function evaluations		
	PSO	SOMA	NMSOMA-M
1	150000	150000	**708**
2	150000	150000	**564**
3	150000	44055	**445**
4	150000	114810	**665**
5	150000	58759	**20874**
6	150000	**72029**	113873
7	150000	150000	**1197**
8	150000	150000	150000
9	150000	150000	**784**
10	150000	150000	**1780**
11	150000	104382	**600**
12	150000	66735	**774**
13	150000	81890	**482**
14	150000	**87777**	135741
15	150000	67296	**925**

Tables 8, 9 and 10, present the mean objective function value corresponding to NMSOMA-M, PSO and SOMA for dimensions 30, 50 and 100 respectively. NMSOMA-M is not only achieving good success rate with lesser function evaluations but also attained objective function value with good accuracy. Results show

Table 8 Mean objective function value for dimension 30

Problem number	Mean objective function value		
	PSO	SOMA	NMSOMA-M
1	0.00971	0.000905	**0.000816**
2	0.00938	0.000817	**0.000572**
3	0.00966	0.000965	**0.000854**
4	0.00936	0.000760	**0.000736**
5	0.00941	0.00708	**0.000968**
6	0.00712	**0.000697**	0.00702
7	36.089	7.169	**0.000670**
8	55.38	327.498	**25.719**
9	0.00983	0.00841	**0.000659**
10	0.00894	0.00861	**0.000562**
11	0.00956	0.00872	**0.000690**
12	0.00943	0.000839	**0.000428**
13	0.00954	0.000807	**0.000779**
14	2618.8	0.00857	**0.000736**
15	0.00954	0.00742	**0.000570**

Table 9 Mean objective function value for dimension 50

Problem number	Mean objective function value		
	PSO	SOMA	NMSOMA-M
1	16.3998	0.000971	**0.000660**
2	0.72846	0.000772	**0.000441**
3	0.00967	0.00926	**0.000445**
4	0.00990	0.000924	**0.000630**
5	3.50605	0.000897	**0.000813**
6	0.09137	**0.000785**	0.00936
7	95.1538	14.249	**0.000737**
8	135.189	193.262	**45.619**
9	59.0000	0.000858	**0.000596**
10	7.27434	0.03268	**0.000574**
11	0.00914	0.00848	**0.000497**
12	0.00968	0.000941	**0.000648**
13	0.00944	0.000785	**0.000510**
14	1904.80	0.00846	**0.000803**
15	0.00963	0.00789	**0.000730**

that the ranking of all the algorithms is PSO $<$ SOMA $<$ NMSOMA-M. Hence NMSOMA-M is most accurate.

Table 11 has presented the results taken by only NMSOMA-M for dimension 1000. Since the performance of SOMA and PSO has not been found satisfactory

Table 10 Mean objective function value for dimension 100

Problem number	Mean objective function value		
	PSO	SOMA	NMSOMA-M
1	19.9251	3.03138	**0.000859**
2	36.1656	0.78662	**0.000270**
3	0.99999	0.00883	**0.000762**
4	669.999	0.00412	**0.000755**
5	5.69121	0.00230	**0.000980**
6	0.00958	**0.000405**	0.00944
7	329.632	55.263	**0.000685**
8	195.04	617.418	**95.352**
9	171.254	0.89232	**0.000711**
10	1912.25	0.14871	**0.00769**
11	*	0.00692	**0.000240**
12	0.00980	0.000816	**0.000742**
13	1534.60	0.000904	**0.000134**
14	*	0.00807	**0.00101**
15	*	0.00873	**0.000730**

rather disappointing as the dimension rises to 1000, results are not taken by these algorithms for dimension 1000. Although NMSOMA-M has already proved its robustness by solving 100 dimensional problems using 10 population size, the main purpose of using NMSOMA-M for solving 1000 dimensional problems is to show its efficacy to solve large scale problems. In Table 11, success rate, average function evaluations and mean objective function value of NMSOMA-M for dimension 1000 has been presented. Success rate obtained by NMSOMA-M is very good. Function evaluations taken by this algorithms is also very less with desirable accuracy. Therefore, NMSOMA-M can be considered as a good approach for solving large scale function optimization problems.

The problems which could not be solved by the particular algorithm is given the sym-bol $(*)$ at the corresponding entries. After analyzing the performance of all three algorithms in terms of three criterions, a compact view of results is reported in Table 12. NMSOMA-M outperforms PSO and SOMA in all the factors considered.

In order to reconfirm our results and to show the results graphically, the relative performance of all the algorithms has been analyzed by using a Performance Index (PI) The relative performance of an algorithm using this PI is calculated by using the following equation (5).

$$PI = \frac{1}{N_p} \sum_{i=1}^{N_p} \left(k_1 \alpha_1^i + k_2 \alpha_2^i + k_3 \alpha_3^i \right) \tag{5}$$

Where

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

Table 11 Performance of NMSOMA-M for dimension 1000

Problem number	Success rate	Average number of function calls	Mean objective function value
1	30	758	0.000645
2	30	581	0.000658
3	30	522	0.000479
4	30	718	0.000537
5	30	54373	0.000945
6	*	*	*
7	25	2123	0.000316
8	30	150000	994.667
9	30	1066	0.000760
10	30	3932	0.000763
11	30	749	0.000651
12	30	660	0.000449
13	30	698	0.000597
14	*	*	*
15	30	1751	0.000570

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

$$\alpha_3^i = \begin{cases} \frac{Mf^i}{Af^i} & if \quad Sr^i > 0 \\ 0 & if \quad Sr^i = 0 \end{cases}$$

Sr^i= Number of successful runs of i^{th} 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
Af^i= Average number of function evaluations of successful runs used by an algorithm in obtaining the solution of i^{th} problem
Mf^i= Minimum of average number of function evaluations of successful runs used by all algorithms in obtaining the solution of i^{th} problem
N_p= Total number of problems analyzed

The variables k_1, k_2 and k_3; $k_1 + k_2 + k_3 = 1$ and $0 \leq k_1, k_2, k_3 \leq 1$ are the weights assigned to percentage of success, mean objective function value and average number of function evaluations 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

Table 12 Comparison of NMSOMA-M, PSO, SOMA

Dimension	Factors	Performance of NMSOMA-M	NMSOMA-M Vs PSO	NMSOMA-M Vs SOMA	Overall performance of NMSOMA-M, PSO and SOMA
30	Success rate	Better	11	06	PSO: 02
		Equal	03	08	SOMA: 09
		Worse	01	01	NMSOMA-M: 13
	Average function calls	Better	15	12	PSO: 0
		Equal	00	01	SOMA: 02
		Worse	00	02	NMSOMA-M: 12
	Mean function value	Better	15	14	PSO: 0
		Equal	00	00	MOMA 01
		Worse	15	14	NMSOMA-M: 14
50	Success rate	Better	13	05	PSO: 01
		Equal	01	09	SOMA: 10
		Worse	01	01	NMSOMA-M: 13
	Average function calls	Better	15	12	PSO: 0
		Equal	00	01	SOMA: 02
		Worse	00	02	NMSOMA-M: 12
	Mean function value	Better	14	14	PSO: 0
		Equal	01	00	MOMA 01
		Worse	00	01	NMSOMA-M: 14
100	Success rate	Better	14	09	PSO: 01
		Equal	01	05	SOMA: 05
		Worse	00	01	NMSOMA-M: 13
	Average function calls	Better	15	12	PSO: 0
		Equal	00	01	SOMA: 02
		Worse	00	02	NMSOMA-M: 12
	Mean function value	Better	15	14	PSO: 0
		Equal	00	00	MOMA 01
		Worse	00	01	NMSOMA-M: 14

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 graph corresponding to each of case (i), (ii) and (iii) for dimension 30 is shown in Figure 3, where the horizontal axis represents the weight *w* and the vertical axis represents the performance index PI. The graph corresponding to each of case (i), (ii) and (iii) for dimension 50 is shown in Figure 4 whereas Figure 5 dipcts the graph corresponding to each of case (i), (ii) and (iii) for dimension 100.

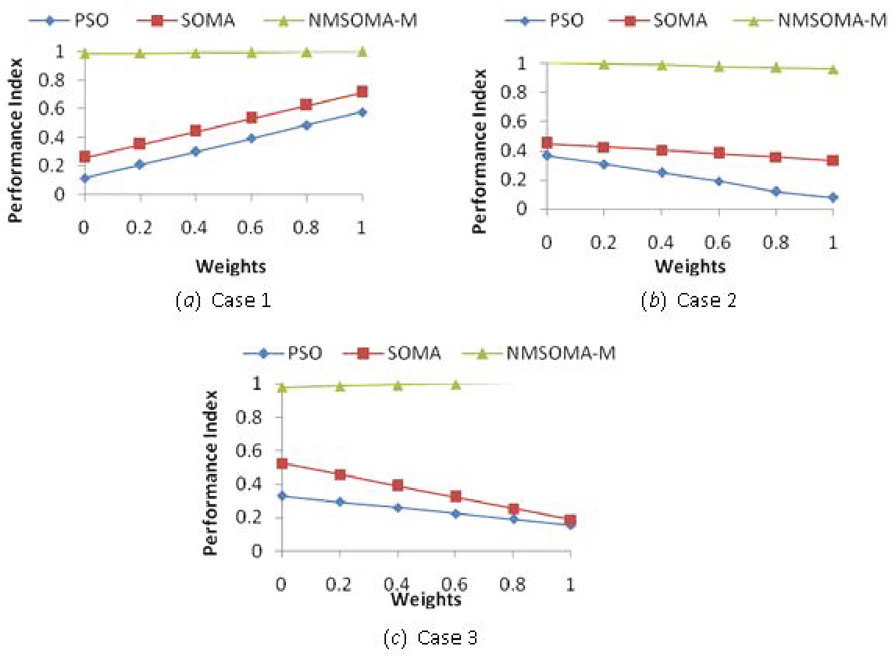

Fig. 3 Performance index of PSO, SOMA and NMSOMA-M for dimension 30

In case (i), the mean objective function value and average number of function evaluations of successful runs are given equal weights. Performance index’s of NMSOMA-M, PSO and SOMA are superimposed in the Figures 3(i), 4(i) and 5(i). It is observed that the value of performance index for NMSOMA-M is more than PSO and SOMA. In case (ii), equal weights are assigned to the numbers of successful runs and mean objective function value of successful runs. Performance indexs of NMSOMA-M, PSO and SOMA are superimposed in the Figures 3(ii), 4(ii) and 5(ii). It is observed that the value of PI for NMSOMA-M is more as compared to PSO and SOMA. Similar case also observed in case (iii). This can be viewed from the Figures 3(iii), 4(iii) and 5(iii).

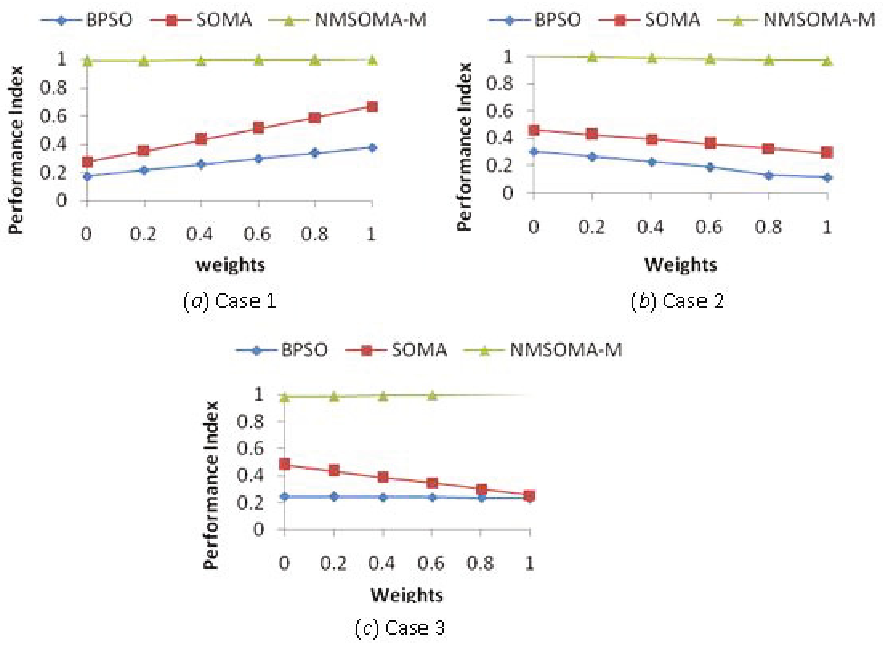

Fig. 4 Performance index of PSO, SOMA and NMSOMA-M for dimension 50

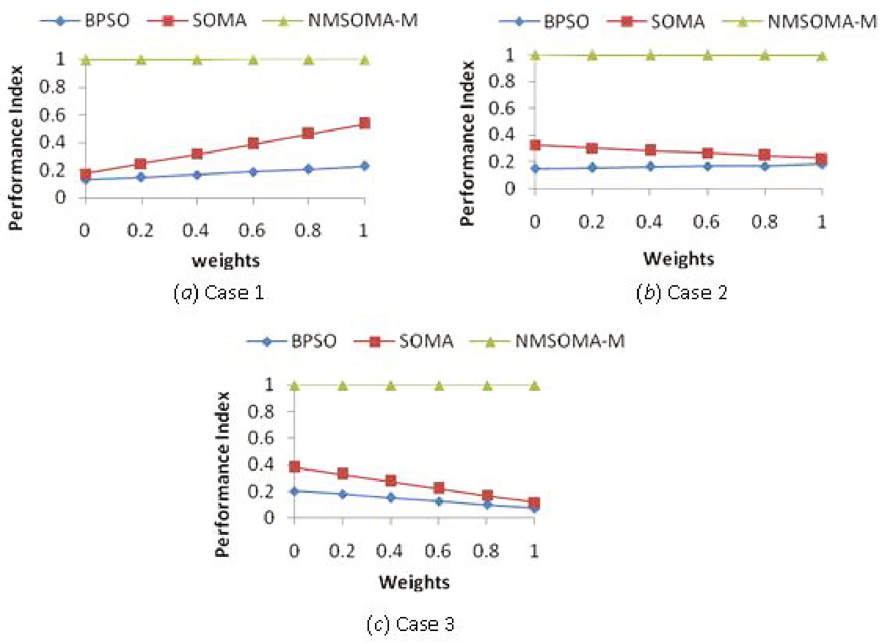

Fig. 5 Performance index of PSO, SOMA and NMSOMA-M for dimension 100

6 Conclusion

In this chapter a hybridized variant NMSOMA-M of SOMA with Nelder Mead crossover operator and log logistic mutation has been presented. This algorithm has been designed to improve the efficiency of SOMA and to overcome the difficulty of premature convergence due to trapping in local optimal solution. Though SOMA works well for solving small scale problems but its performance becomes poor due to loss of diversity as the dimension of the solution space becomes large. In the working of presented algorithm NMSOMA-M, NM crossover and log logistic mutation operator are used to maintain the diversity in the solution space by creating new points. NMSOMA-M has been tested on a set of 15 scalable test problems and results are taken for dimension 30, 50, 100 and 1000 respectively. Since the performance of PSO and SOMA was not found satisfactory, the results obtained by NMSOMA-M have been compared with results obtained by SOMA and PSO only for dimension 30, 50 and 100. NMSOMA-M not only attained good success rate, in less function evaluations with desirable accuracy but also use very small population size to work with. At last on the basis of the results presented it can be concluded that the presented algorithm NMSOMA-M is an efficient, reliable and accurate to solve large scale real life optimization problems.

References

1. Zelinka, I., Lampinen, J.: SOMA - Self organizing migrating algorithm. In: Proceedings of the 6th International Mendel Conference on Soft Computing, Brno, Czech, Republic, pp. 177–187 (2000)
2. Domingo, O.B., Cessae, H.M., Nicolas, G.P.: CIXL2: A crossover operator for evolutionary algorithms based on research. Journal of Artificial Intelligence Research 24, 1–48 (2005)
3. Chelouah, R., Siarry, P.: A hybrid method combining continuous tabu search and nelder – Mead simplex algorithm for global optimization of multiminima functions. European Journal of Operational Research 161, 636–654 (2005)
4. Deep, K., Dipti, S.: A new hybrid self organizing migrating genetic algorithm for function optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2796–2803 (2007)
5. Fan, K.-S., Liang, Y.C., Zahara, E.: A genetic algorithm and a particle swarm optimizer hybridized with nelder-mead simplex search. Computers and Industrial Engineering 50, 401–425 (2006)
6. Premalatha, K., Nataranjan, A.M.: Hybrid PSO and GA for global optimization. International Journal of Open Problems and Computational Mathematics 2(4), 597–608 (2009)
7. Khosravi, A., Lari, A., Addeh, J.: A new hybrid of evolutionary and conventional optimization algorithm. Applied Mathematical Sciences 6, 815–825 (2012)
8. Ghatei, S., Panahi, F.T., Hosseinzadeh, M., Rouhi, M., Rezazadeh, I., Naebi, A., Gahtei, Z., Khajei, R.P.: A new hybrid algorithm for optimization using PSO and GDA. Journal of Basic and Applied Scientific Research 2, 2336–2341 (2012)

9. Esmin, A.A.A., Matwin, S.: A hybrid particle swarm optimization algorithm with genetic mutation. International Journal of Innovative Computing, Information and Control 9, 1919–1934 (2013)
10. Pant, M., Thangaraj, R., Abraham, A.: A new PSO algorithm with crossover operator for global optimization problems. In: Corchado, E., Corchado, J.M., Abraham, A. (eds.) Innovations in Hybrid Intelligent Systems, vol. 44, pp. 215–222. Springer, Heidelberg (2007)
11. Bansal, J.C., Deep, K.: Quadratic approximation PSO for economic dispatch problems with valve-point effects. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) SEMCCO 2010. LNCS, vol. 6466, pp. 460–467. Springer, Heidelberg (2010)
12. Deep, K., Thakur, M.: A new mutation operator for real coded genetic algrithms. Applied Mathematics and computation 193, 229–247 (2007)
13. Xing, L.N., Chen, Y.W., Yang, K.W.: A novel mutation operator base on immunity operation. European Journal of Operational Research 197, 830–833 (2009)
14. Deep, K., Shashi, Katiyar, V.K.: A new real coded genetic algorithm operator: Log logistic mutation. In: Deep, K., Nagar, A., Pant, M., Bansal, J.C. (eds.) Proceedings of the International Conference on SocProS 2011. AISC, vol. 130, pp. 193–200. Springer, Heidelberg (2012)
15. Mohan, C., Shankar, K.: Random search technique for global optimization. Asia Pacific Journal of Operations Research 11, 93–101 (1994)
16. Deep, K., Das, K.N.: Quadratic approximation based hybrid genetic algorithm function optimization. Applied Mathematics and Computations 203, 86–98 (2008)
17. Deep, K., Bansal, J.C.: Hybridization of particle swarm optimization with quadratic approximation. Opsearch 46, 3–24 (2009)
18. Deep, K., Das, K.N.: Performance improvement of real coded genetic algorithm with quadratic approximation based hybridization. International Journal of Intelligent Defence Support Systems 2, 319–334 (2010)
19. Pant, M., Thangaraj, R., Abraham, A.: A new quantum behaved particle swarm optimization. In: Keijzer, M. (ed.) Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO, Netherlands, pp. 87–94 (2008)
20. Singh, D., Agrawal, S., Singh, N.: A novel variant of self organizing migrating algorithm for function optimization. In: Pant, M., Deep, K., Nagar, A., Bansal, J.C. (eds.) Proceedings of the Third International Conference on Soft Computing for Problem Solving. AISC, vol. 258, pp. 225–234. Springer, Heidelberg (2014)
21. Eberhart, R.C., Kennedy, J.: Partcle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
22. Zelinka, I.: SOMA - Self organizing migrating algorithm. In: Onwubolu, G.C., Babu, B.V. (eds.) New Optimization Techniques in Engineering. STUDFUZZ, vol. 141, pp. 167–217. Springer, Heidelberg (2004)
23. Spendley, W., Hext, G.R., Himsworth, F.R.: Sequential application of simplex designs in optimization and evolutionary operation. Technometrics 4, 441–461 (1962)