01 Sep 2008

Empirical Study of a Hybrid Algorithm Based on Clonal Selection and Small Population Based PSO.

Pinaki Mitra

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

2008 IEEE Swarm Intelligence Symposium
St. Louis MO USA, September 21-23, 2008

Empirical Study of a Hybrid Algorithm Based on Clonal Selection and Small Population Based PSO

Pinaki Mitra, *Student Member, IEEE*, Ganesh K. Venayagamoorthy, *Senior Member, IEEE*

Abstract— **In this paper, a hybrid algorithm, based on Clonal Selection Algorithm (CSA) and Small Population Based Particle Swarm Optimization (SPPSO) is introduced. The performance of this new algorithm (CS^2P^2SO) is observed for four well known benchmark functions. The SPPSO is a variant of conventional PSO (CPSO), introduced by the second author of this paper, where a very small number of initial particles are used and after a few iterations, the best particle is kept and the rest are replaced by the same number of regenerated particles. On the other hand, CSA belongs to the family of Artificial Immune System (AIS). It is an evolutionary algorithm, where, during evolution, the antibodies which can recognize the antigens proliferate by cloning. With the hybridization of these two algorithms, the strength of CPSO is enhanced to a great extent. The concept of SPPSO helps to find the optimum solution with less memory requirement and the concept of CSA increases the exploration capability and reduces the chances of convergence to local minima. The test results show that CS^2P^2SO performs better than CPSO and SPPSO for the Sphere, Rosenbrock's, Rastrigin's and Griewank's functions.**

I. Introduction

PARTICLE Swarm Optimization (PSO) has been shown to have great potential for solving single and multi-objective optimization problems [1]. It is a simple, flexible and well balanced algorithm for carrying out local and global search processes. Here, a group of particles, called a swarm, move in a multi-dimensional search space to find out the global best solution. As the number of particles in the swarm increases, the convergence to a global solution is more and more ensured. The reason is, higher the number of particles, the greater the exploration of the search space. But, as the number of particles increases, the memory requirement for the algorithm also increases which is often not permissible in the real world application of the algorithm with digital signal processors or microcontrollers, etc. Again, if within the first few iterations, one of the particles moves very close to a local minima and none are close to the global minima, there is a chance that the entire swarm is misguided to converge to that local minima. This kind of situation frequently happens for the functions with a large number of local minima. In order to get rid of these two problems, SPPSO algorithm was proposed in [2] and [3]. The concept of SPPSO is to start with a small number of particles and after a few iterations, replace all the particles except the global best with same number of regenerated particles. In this method, since the PSO runs with a very small number of particles, the memory requirement is reduced a lot. Also, since after few iteration a new set of particles are introduced, the chance of fixation to a local minima decreases.

To further improve the exploration capability of the SPPSO algorithm, which is very much essential for the functions having multiple closely-located local minima, this paper proposes a modified version of SPPSO in conjunction with the Clonal Selection Algorithm (CSA). CSA belongs to the family of Artificial Immune System (AIS). AIS is a computational intelligence paradigm inspired by the natural immune system of human body [4]. Cloning and mutation are the two most vital steps of CSA, which make the exploration potential of CSA very high. Few attempts have already been made by the researchers to amalgamate PSO with CSA. In [4], CSA is used on the global best particles of a certain number of generations stored in memory. Therefore, this process also needs a large memory. In [5], a target oriented mutation is proposed, where the mutation process resembles the velocity update equation of PSO. This process is rather a variant of CSA and is not related to the swarm intelligence techniques directly. In [6], a hybrid algorithm was proposed, where half of the population was going through a position and velocity update process like the PSO and rest half of the population was following CSA simultaneously. But, in this paper, CSA is used during the regeneration process of a modified SPPSO algorithm. With this application, the memory requirements as well as the chances of convergence to local minima are reduced to a large extent. The proposed algorithm is referred to as CS^2P^2SO by the authors.

The rest of the paper is organized as follows: Section II describes the PSO, SPPSO and the proposed CS^2P^2SO algorithms in detail. Section III describes the benchmark functions used in this paper. The experimental settings are presented in Section IV. A comparative study of the performances of these three algorithms is presented in Section V. Finally, conclusions are drawn in Section VI.

Manuscript received June 15, 2008. This work was supported in part by the National Science Foundation, USA under the NSF CAREER grant # ECCS 0348221.

Pinaki Mitra is with Real-Time Power and Intelligent Systems Laboratory, ECE Department, Missouri University of Science and Technology, Rolla, MO 65401, USA (phone: 609-384-1302; e-mail: pm33d@mst.edu).

Ganesh K. Venayagamoorthy is with Real-Time Power and Intelligent Systems Laboratory, ECE Department, Missouri University of Science and Technology, Rolla, MO 65401, USA (e-mail: gkumar@ieee.org).

978-1-4244-2705-5/08/$25.00 ©2008 IEEE

II. THREE VARIANTS OF PSO

A. *Conventional Particle Swarm Optimization (CPSO)*

Particle swarm optimization is a population based search algorithm which aims to replicate the motion of flock of birds and school of fishes [7], [8]. A swarm is considered to be a collection of particles, where each particle represents a potential solution to the problem. The particle changes its position within the swarm based on the experience and knowledge of its neighbors. Basically it 'flies' over the search space to find the optimal solution [8], [9].

Initially a population of random solutions is considered. A random velocity is also assigned to each individual particle with which they start flying within the search space. Also, each particle has a memory which keeps track of the previous best position of the particle and the corresponding fitness. This previous best value is called 'p_{best}'. There is another value called 'g_{best}', which is the best value of all the 'p_{best}' values of the particles in the swarm. The fundamental concept of the PSO technique is that the particles always accelerate towards their 'p_{best}' and 'g_{best}' positions at each time step. Fig. 1 demonstrates the concept of PSO where,

a) $x_{id}(k)$ is the current position of i^{th} particle with d dimensions at instant k.
b) $x_{id}(k+1)$ is the position of i^{th} particle with d dimensions at instant $(k+1)$.
c) $v_{id}(k)$ is the initial velocity of the i^{th} particle with d dimensions at instant k.
d) $v_{id}(k+1)$ is the initial velocity of the i^{th} particle with d dimensions at instant $(k+1)$.
e) w is the inertia weight which stands for the tendency of the particle to maintain its previous position.
f) c_1 is the cognitive acceleration constant, which stands for the particles' tendency to move towards its 'p_{best}' position.
g) c_2 is the social acceleration constant which represents the tendency of the particle to move towards the 'g_{best}' position.

The velocity and the position of the particle are updated according to the following equations. The velocity of the i^{th} particle of d dimension is given by:

$$v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \cdot rand_1 \cdot (p_{best\_id}(k) - x_{id}(k)) + c_2 \cdot rand_2 \cdot (g_{best\_id}(k) - x_{id}(k)) \quad (1)$$

The position vector of the i^{th} particle of d dimension is updated as follows:

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (2)$$

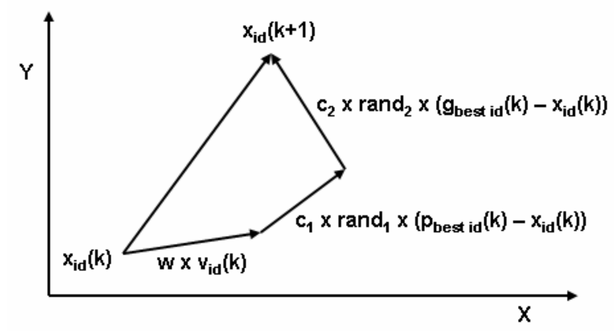

Fig.1. Concept of changing a particle's position in two dimension [10]

B. *Small Population Based Particle Swarm Optimization (SPPSO)*

It is a variant of PSO algorithm. This is different from CPSO in two respects. First, a very small number of initial populations compared to CPSO are used in this algorithm. The number of particles in SPPSO can be below 5. Secondly, after I number of iterations, all the particles except the *gbest* particle are replaced by randomly generated new particles. The *pbest* positions are also retained and are carried over to the next I iterations. Since after each I iterations, new set of particles are generated, this algorithm behaves as good as a large population based PSO with a very small memory requirement.

In this paper, a modification is introduced in the concept of SPPSO. If only the gbest particle is taken from the first I iterations, the potential of the other particles having high fitness value are ignored. To overcome this, instead of taking only the gbest particle from the first I iterations, half of the total number of particles are sorted out based on their fitness and are carried over to the next I iterations. The rest half having lower fitness values are replaced by fresh random particles.

C. *Clonal Selection Based SPPSO (CS^2P^2SO)*

To further enhance the exploration potential of SPPSO, CSA is used. Clonal Selection Algorithm (CSA) is an integrated part of AIS, which explains how an immune response is mounted when a non-self antigen is recognized by the B cells [11]. It is an evolutionary algorithm, where, during evolution, the antibodies which can recognize the antigens proliferate by cloning [12]. The term 'fitness' is equivalent to 'affinity' in AIS.

The general steps involved in CSA are as follows:

- To create a population P of random solutions to the given problem.
- To evaluate the fitness of each member.
- To rank the population by fitness.
- While termination condition not met:
  - To take the fittest N population members.
  - To create n clones from each member of N, where n is proportional to the fitness of the member of N.
  - To evaluate the fitness of the cloned members of N.

- To mutate each clone inversely proportionally to its fitness.
- Given P and the mutated clones, to choose the best P members and form a new population.

In the CS^2P^2SO algorithm proposed in this paper, CSA is applied on the best fit particles selected after I iterations of PSO. If the initial population size is P, then CSA is applied on $N=P/2$ best fit particles. The number of clones generated is given by the following equation:

$$N_c = \sum_{i=1}^{N} round\left(\frac{B*N}{i}\right) \tag{3}$$

Where

N = Total number of particles to be cloned. In terms of CSA it is the number of antibodies.

B = Cloning index. By varying this parameter, number of clones can be regulated.

i = 1 for highest fitness. In terms of CSA it is the highest affinity. The second highest affinity is 2 and so on.

N_c = The entire population size after cloning.

This new set C of N_c number of cloned antibodies are then put through a mutation process in such a way that the best fit clone will have least mutation. This is done by setting a mutation rate (α) for each clone which is given by the following equation:

$$\alpha = \exp(-\rho) \tag{4}$$

where, ρ is the affinity of that clone. In this paper, the target oriented mutation as described in [5] is adopted, which is represented by the following equation:

$$C^* = C + \alpha * rand * C + \alpha * rand * (C - gbest) \tag{5}$$

In the above equation

C^* = mutated version of the clone C of set C.

These mutated clones form a new set termed as set C^*. The cloning and mutation applied on the best fit particles of PSO increases the exploration potential of the algorithm near the vicinity of the fittest particles and distant regions from the less fit particles in the search space.

After cloning and mutation, the concept of regeneration of SPPSO is adopted. $P/2$ numbers of randomly generated new particles are now added to the elements of set C^*. $P/2$ best fit particles obtained in the first I iterations are also added to the set C^*. Therefore, the number of elements of set C^* becomes $N_c+P/2+P/2 = N_c+P$.

Now, from those N_c+P number of elements of set C^*, the fittest P particles are reselected for the next set of I iterations. The entire process is repeated until any termination criteria are met. The flowchart of the entire process is shown in Fig. 2.

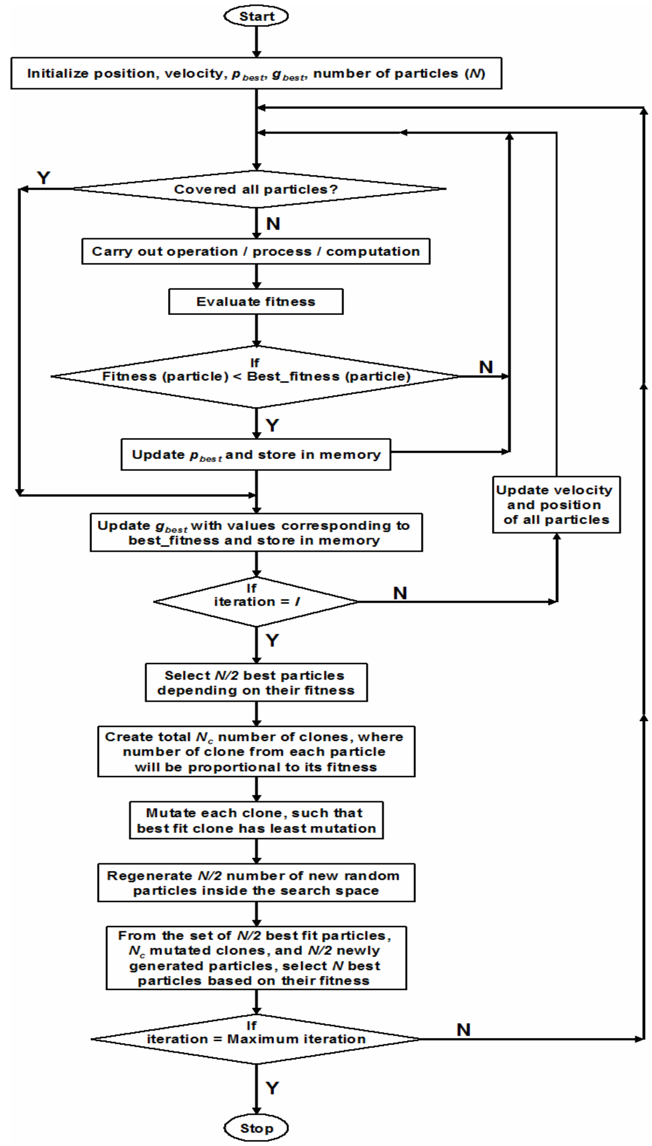

Fig.2. Flowchart of CS^2P^2SO Algorithm

III. BENCHMARK FUNCTIONS

In this paper, four benchmark functions are used for optimization problem. The functions are: Sphere function, Rosenbrock's function, Rastrigin's function and Griewank's function. The mathematical expressions for the functions are as follows:

1) Sphere function:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \qquad -5.12 \le x_i \le 5.12 \tag{6}$$

2) Rosenbrock's function:

$$f_2(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \qquad -2.048 \le x_i \le 2.048 \tag{7}$$

3) Rastrigin's function:

$$f_3(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \boldsymbol{\pi} \cdot x_i)) \qquad (8)$$

$$-5.12 \le x_i \le 5.12$$

4) Griewank's function:

$$f_4(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (9)$$

$$-600 \le x_i \le 600$$

IV. Experimental Settings

For each benchmark functions, there is an upper and a lower limit to each element as shown in (6) – (9). The position and velocities of the particles are initialized symmetrically within that limit. The maximum and minimum values of the position and velocity are also constrained within the same limit. The initial range of particle position and velocities and their limits are shown in TABLE I.

In this paper, to study the performance of CPSO, SPPSO and CS^2P^2SO algorithms, the population size of 4, 10, 20 and 40 are examined. The small value of population size like 4 and 10 are taken to observe the effectiveness of the small population based algorithms like SPPSO and CS^2P^2SO. For each population size, the dimension of 5, 10, 20, 30 and 50 are considered. As the dimension of the problem increases, the number of iterations is also increased proportionally. The number of iterations corresponding to the dimensions is given in TABLE II.

TABLE I

Initial Range and Upper and Lower Limits of Particle Positions and Velocities

Functions	Initial range and upper and lower limits of particle positions and velocities
f_1	-5.12 / 5.12
f_2	-2.048 / 2.048
f_3	-5.12 / 5.12
f_4	-600 / 600

TABLE II

Number of Iterations Corresponding to Number of Dimensions

Dimension	Iterations
5	500
10	1000
20	2000
30	3000
50	5000

V. Results

In this paper, CPSO, SPPSO and CS^2P^2SO algorithms are used to find out the minimum of the four benchmark functions. All those four benchmark functions have their minimum values equal to zero. The constants w, c_1 and c_2 of the algorithm are taken respectively as 0.8, 2 and 2. The cloning index (B) of CS^2P^2SO algorithm is taken equal to 1. For each population size and dimension, all the algorithms are run for 50 times and the mean and standard deviation of minimal fitness for those 50 independent runs are presented in TABLE III – VI. The value 0 in the tables stands for 1×10^{-323}, since the performance is studied in Matlab.

A. *Sphere Function*

TABLE III shows the results obtained with the three algorithms for the sphere function. It is observed that for CPSO algorithm, the performance becomes poorer for smaller population size. Also, for a fixed population size, as the dimension of the problem increases, the result deteriorates. The same trend is observed for SPPSO with small population size like 4 and 10. But with higher population size, the trend is reversed. With population size 20 and 40, with the increasing number of dimension, the performance of SPPSO improves. This is because, as the number of particles increases, the performance of the SPPSO algorithm is more dependent on the number of iterations than the number of dimensions. Same is the case with CS^2P^2SO. But this trend is observed for all population sizes of CS^2P^2SO. Also, with very small population size, CS^2P^2SO reaches too close to the global minima compared to the other two algorithms. As a whole, the performance of CS^2P^2SO is much better than CPSO and SPPSO. Though, it is worth mentioning that the performance of SPPSO is quite better than CPSO algorithm.

B. *Rosenbrock's Function*

TABLE IV shows the results obtained with the three algorithms for Rosenbrock's function. Among all the four functions tested in this paper, this is the toughest one to optimize. So, none of the three could reach 0 for this function. A similar trend is observed for all the functions. With the increase in dimension the performance is deteriorated for a fixed population size and for a fixed value of dimension, the performance improves with larger population size. But as a whole, the CS^2P^2SO reaches more close to the global minima compared to the other two algorithms and just like the sphere function, the overall performance of SPPSO is better than CPSO algorithm.

C. *Rastrigin's Function*

The results for Rastrigin's function with these three algorithms are presented in TABLE V. In case of Rastrigin's function again, the performance of all the three algorithms becomes poorer with the increase in dimension of the problem. With smaller dimension like 5 and 10, only SPPSO and CS^2P^2SO could reach the global minima. Whereas, CPSO fails to reach the global minima even with large number of particles. It is also found that the performance of all the algorithms improves with the increase in the number of particles. Here again, the proposed CS^2P^2SO algorithm defeats the performance of the other two as a whole.

D. *Griewank's Function*

For Griewank's function, the proposed CS^2P^2SO

algorithm outperforms the other two algorithms. TABLE VI shows that, except for the first case, the CS^2P^2SO could find out the global minima in all the other cases. SPPSO also performed much better than CPSO and with larger population size it also could reach the global minima in many cases. But with small population size like 4 and 10, it has shown the similar trend like CPSO, where the increase in dimension deteriorated its performance.

From all the results, it is clearly observed that CS^2P^2SO is a superior algorithm with respect to CPSO and SPPSO and SPPSO is again much more efficient than the CPSO algorithm.

TABLE III
EMPIRICAL EVALUATION WITH SPHERE FUNCTION

Pop	Dim	Iter	CPSO	SPPSO	CS^2P^2SO
4	5	500	0.0358±0.0923	6.2745e-004±0.0022	5.2257e-094±3.4848e-093
	10	1000	0.3535±0.6114	0.0141±0.0267	1.2432e-194±0
	20	2000	2.0759±3.3189	0.0672±0.0997	0±0
	30	3000	3.8765±5.4501	0.1252±0.2054	0±0
	50	5000	8.1027±9.6008	0.1629±0.2656	0±0
10	5	500	6.0695e-006±1.3267e-005	1.2035e-008±8.5103e-008	1.9445e-115±1.0663e-114
	10	1000	0.0061±0.0104	9.8435e-007±4.5211e-006	2.7056e-221±0
	20	2000	0.0688±0.0619	2.9281e-005±1.0830e-004	0±0
	30	3000	0.1062±0.1246	1.0053e-004±2.008e-4	0±0
	50	5000	0.1748±0.1930	7.1247e-004±0.0015	0±0
20	5	500	7.5677e-009±2.6258e-008	3.0119e-017±1.1224e-016	5.4997e-140±2.0788e-139
	10	1000	2.4631e-005±4.5470e-005	6.005e-092±2.1055e-089	1.0631e-238±0
	20	2000	0.0028±0.0034	0±0	0±0
	30	3000	0.0072±0.0142	0±0	0±0
	50	5000	0.0175±0.0264	0±0	0±0
40	5	500	9.1267e-013±5.3383e-012	4.4088e-043±2.0986e-040	7.3079e-173±0
	10	1000	3.0231e-010±7.9719e-010	1.9116e-137±7.0016e-131	0±0
	20	2000	3.0788e-005±1.0225e-004	0±0	0±0
	30	3000	1.3041e-004±3.0829e-004	0±0	0±0
	50	5000	0.0013±0.0022	0±0	0±0

TABLE IV
EMPIRICAL EVALUATION WITH ROSENBROCK'S FUNCTION

Pop	Dim	Iter	CPSO	SPPSO	CS^2P^2SO
4	5	500	3.573±1.2388	2.1004±1.2784	0.6236±1.3384
	10	1000	13.2878±8.6942	8.8783±1.2904	5.7806±0.8585
	20	2000	80.6819±159.5915	21.5738±13.1241	16.3986±1.2930
	30	3000	212.9937±379.0389	34.4495±26.5418	38.5545±60.1895
	50	5000	147.9860±134.8622	51.8723±6.2789	46.7094±1.0035
10	5	500	0.8605±1.1513	0.1418±0.5352	0.0028±0.0174
	10	1000	8.0107±1.4503	6.8367±0.8227	2.4736±0.6016
	20	2000	19.4184±1.7785	18.0174±0.5327	13.4540±1.0880
	30	3000	30.7021±2.5543	28.2339±0.4855	23.4158±0.9667
	50	5000	61.4223±59.4165	48.2705±0.3529	47.1548±25.3963
20	5	500	0.1051±0.5340	0.0031±0.0075	3.1651e-006±1.0331e-005
	10	1000	6.0174±1.7675	3.8874±0.8674	0.1561±0.1546
	20	2000	18.0331±0.7738	17.0026±0.7226	10.9298±0.8772
	30	3000	28.3921±0.7609	27.6723±0.4342	20.8543±0.7007
	50	5000	48.5627±0.6230	47.9326±0.4867	40.4494±0.7457
40	5	500	4.6846e-006±1.2703e-005	5.2083e-005±1.1492e-004	1.4730e-007±4.6889e-007
	10	1000	1.3938±.9597	0.7910±0.4610	4.1489e-004±5.2973e-004
	20	2000	15.6658±0.9161	14.5996±0.7987	7.4380±0.6659
	30	3000	27.1107±0.9077	26.1223±0.5583	17.5123±0.6171
	50	5000	47.4902±0.7140	47.1161±0.4736	36.0883±5.2433

TABLE V
EMPIRICAL EVALUATION WITH RASTRIGIN'S FUNCTION

Pop	Dim	Iter	CPSO	SPPSO	CS^2P^2SO
4	5	500	5.8375±6.7676	7.3452e-004±0.0046	0±0
	10	1000	22.4120±13.8968	5.0709±10.0228	0±0
	20	2000	64.8506±34.5624	17.7888±16.5961	2.5807±10.7953
	30	3000	109.9306±50.6274	46.0021±48.1055	41.2974±51.7347
	50	5000	209.3168±85.5259	77.2140±71.9877	65.3541±67.7070
10	5	500	1.5346±1.9510	0±0	0±0
	10	1000	8.9340±8.2079	0.1001±0.7079	0±0
	20	2000	28.3200±18.1632	7.6967±13.3970	5.9396±24.0424
	30	3000	45.0533±28.9457	22.5795±25.1284	20.6453±40.8826
	50	5000	86.5298±47.7625	59.6761±58.2144	56.1673±52.0211
20	5	500	0.4852±0.8851	0±0	0±0
	10	1000	4.5907±7.3045	0±0	0±0
	20	2000	13.2754±12.5268	1.9850±6.9772	1.6355±10.3252
	30	3000	33.3506±23.3966	27.8003±29.1451	13.7279±34.6677
	50	5000	59.6130±38.7065	54.4762±59.3441	49.6343±70.8699
40	5	500	0.1602±0.7346	0±0	0±0
	10	1000	0.8368±2.0053	0±0	0±0
	20	2000	12.2509±13.4430	1.0765±5.3408	0±0
	30	3000	16.4441±13.7541	11.1707±17.6943	6.3069±23.6262
	50	5000	40.8122±36.0842	37.7112±54.3281	30.7652±52.6359

TABLE VI
EMPIRICAL EVALUATION WITH GRIEWANK'S FUNCTION

Pop	Dim	Iter	CPSO	SPPSO	CS^2P^2SO
4	5	500	0.4821±0.7154	0.0380±0.0844	4.3379e-012±3.0592e-011
	10	1000	2.7484±3.3059	0.2614±0.3975	0±0
	20	2000	6.4642±7.2139	0.6217±0.6726	0±0
	30	3000	16.1641±20.5914	0.8123±0.9498	0±0
	50	5000	26.6433±36.9285	1.3777±1.5033	0±0
10	5	500	0.1612±0.1909	0±0	0±0
	10	1000	0.4881±0.3428	0±0	0±0
	20	2000	0.9875±.07799	0.0239±0.0937	0±0
	30	3000	1.3824±0.0384	0.0663±0.1583	0±0
	50	5000	1.2783±1.0717	0.0919±0.2027	0±0
20	5	500	0.0411±0.0850	0±0	0±0
	10	1000	0.1526±0.1938	0±0	0±0
	20	2000	0.4806±0.4363	1.9227e-004±0.0014	0±0
	30	3000	0.5568±0.4953	0.0024±0.0168	0±0
	50	5000	0.6222±0.5320	0.0144±0.0585	0±0
40	5	500	0.0111±0.0382	0±0	0±0
	10	1000	0.0524±0.0969	0±0	0±0
	20	2000	0.0667±0.1531	0±0	0±0
	30	3000	0.1132±0.2061	0±0	0±0
	50	5000	0.1762±0.2793	0±0	0±0

VI. CONCLUSION

This paper proposes a hybrid optimization algorithm, CS^2P^2SO, which uses the concept of clonal selection and small population based particle swarm optimization. A modified version of SPPSO algorithm is used in this paper and the exploration capacity is enhanced with the application of cloning and mutation operations. The advantage of the proposed algorithm is that, since it uses a very small population size, the memory requirement is very low. Simultaneously, due to the use of the regeneration concept and the application of clonal selection, the algorithm can easily escape from local minima. The performance of the proposed CS^2P^2SO algorithm is observed for four benchmark functions. Also, its performance is compared with CPSO and SPPSO algorithm. It is found that for all the functions, the CS^2P^2SO outperforms the other two algorithms and in most of the cases reaches very close to the global minima even with a small population size.

ACKNOWLEDGMENT

The authors acknowledge the financial support for this research study provided by the National Science Foundation, USA under the NSF CAREER grant # ECCS 0348221.

REFERENCES

[1] A. Coello Carlos and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," *Proceedings of International Conference on Evolutionary Computation*, pp. 1051-1056, 2002.

[2] T. K. Das, S. R. Jetti, and G. K. Venayagamoorthy, "Optimal Design of SVC Damping Controllers with Wide Area Measurements Using Small Population based PSO", *2006 International Joint Conference on Neural Networks*, pp. 2255-2260, July 16-21, 2006.

[3] T. K. Das and G. K. Venayagamoorthy, "Optimal Design of Power System Stabilizers Using a Small Population Based PSO", *IEEE Power Engineering Society General Meeting, 2006*, pp. 1-7, 18-22 June, 2006.

[4] Y. Tan, Z. M. Xiao, "Clonal Particle Swarm Optimization and Its Application", *2007 IEEE Congress on Evolutionary Computation*, pp. 2303-2309, 2007.

[5] S.A. Panimadai Ramaswamy, G. K. Venayagamoorthy and S.N. Balakrishnan, "Optimal Control of Class of Non-linear Plants Using Artificial Immune Systems: Application of the Clonal Selection Algorithm", *22nd IEEE International Symposium on Intelligent Control*, pp. 249-254, 1-3 October, 2007.

[6] Q. Wang, C. Wang, X. Z. Gao, "A Hybrid Optimization Algorithm Based on Clonal Selection Principle and Particle Swarm Intelligence", *Sixth International Conference on Intelligent Systems Design and Applications*, pp. 975-979, 16-18 Oct, 2006.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proceedings of 1995 IEEE Conference on Neural Networks*, vol. 4, pp. 1942-1948.

[8] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, R. G. Harley, "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems", *IEEE Transactions on Evolutionary Computation*, vol. 12, issue 2, pp. 171-195, April 2008.

[9] S. Mohaghegi, Y. del Velle, G. K. Venayagamoorthy and R. Harley, "A Comparison of PSO and Backpropagation for Training RBF Neural Networks for Identification of a Power System with STATCOM", *Proceedings of IEEE Swarm Intelligence Symposium*, 2005, pp. 381-384.

[10] Yoshida, Y. Fukuyama, S. Takayama and Y. Nakanishi, "A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power System Considering Voltage Security Assessment", *Proceedings of the IEEE SMC'99 Conference*, Vol: 6, pp. 497-502, 1999.

[11] X. Wang, "Clonal Selection Algorithm in Power Filter Optimization", *Proceedings of the 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*, pp. 122-127.

[12] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach,* London, UK: Springer-Verlag, 2002.