

## A NEW APPROACH TO GLOBAL OPTIMIZATION MOTIVATED BY PARLIAMENTARY POLITICAL COMPETITIONS

ALI BORJI

School of Cognitive Sciences  
Institute for Studies in Theoretical Physics and Mathematics  
Niavaran Bldg, P.O. Box 19395-5746, Tehran, Iran  
borji@ipm.ir

MANDANA HAMIDI

Electronic and Computer Engineering Department  
Islamic Azad University Branch of Zarghan  
Azad University Ave, P.O. Box 73415-314, Zarghan, Iran  
mandana.hamidi@gmail.com

Received December 2007; revised May 2008

**ABSTRACT.** *Several biology inspired optimization algorithms such as Genetic Algorithms, Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) have previously been proposed by researchers. Recent approaches in numerical optimization have shifted to motivate from complex human social behaviors. In this paper, a new optimization algorithm, namely parliamentary optimization algorithm (POA) is proposed by studying the competitive and collaborative behaviors of political parties in a parliament. Experimental results reveal that our proposed approach is superior to PSO approach over some benchmark multidimensional functions.*

**Keywords:** Swarm Optimization, Computational Intelligence, Evolutionary Computing, Politics, Political Competitions, Social Competitive Behaviors

1. **Introduction.** Human beings have always been inspired by natural and biological processes for purposes like making tools, seeking foods, protection and transportation. Recently, engineers and specially computer scientists have proposed various nature inspired techniques and computational approaches for solving ever increasingly complex problems in engineering and society. Some of the successful applications of such methods to real-world applications and optimization problems include manufacturing, scheduling, anomaly detection, data mining, engineering design, software testing and bioinformatics.

Scale and complexity of the optimization problems in today's industry demand efficient solutions in terms of generating high quality solutions within a reasonable amount of time. Evolutionary algorithms, ant systems, artificial immune systems, artificial neural networks and swarm intelligence systems are examples of optimization approaches that are motivated by biology and life sciences. They satisfy the above mentioned conditions and have proved to be highly practical in several real world applications.

During the evolution, natural systems have progressively evolved from biological to social systems. They range from simple biological entities like single cell organisms, e.g.

bacteria, to highly complex social systems like primate and human societies. Correspondingly there is a spectrum of biology inspired approaches. In what follows we point briefly to some important instances. Genetic algorithms based on the Darwins theory of evolution are now typical examples of biology inspired optimization methods [1,2,3,4]. Biology of the immune system has led to artificial immune inspired mechanisms with successful applications in computer networks, cryptography, etc. [5,6]. Simple modeling of biological neurons has resulted to artificial neural networks with successful applications in areas like prediction and classification [7]. There are also other methods like molecular computing [8], membrane computing [9] and DNA computing [10] inspired by the lower level biological mechanisms. Methods which imitate the social behaviors of insects or animals like fish schools, birds, bees, ants, etc. are together called swarm based approaches. Example methods in this category are particle swarm optimization (PSO) [11,12,13,14] and ant colony optimization (ACO) algorithms [15,16]. ACO solves optimization problems by moving tracks of ants with the assistance of pheromone. PSO finds the solutions by moving the particles in the solution space on the balance of personal and best group experiences. Methods that imitate the social interactions among humans, as the most intelligent creatures on the earth, are the latest approaches. In this study we investigate competition/coordination/cooperation interactions among humans as a source of socially inspired computational intelligence. Some other methods and applications inspired by social behavior of humans and animals have also been proposed [17,18,19].

Traditional methods are more appropriate when scale of the problem is small and prior knowledge about it is already available. Nature inspired optimization methods on the other hand are useful when little is known about the underlying problem. This class of methods is becoming increasingly more important in today's complex and demanding applications.

Our aim in this paper is to introduce a new optimization algorithm motivated from human social behaviors in political environments. Particularly competitions among political groups during elections for the chairman position have been our main inspiration source for formulating a new method for global optimization.

The rest of this paper is organized as follows. In section two, political competitions to the extent relevant to this work as well as our proposed algorithm are described. It is then followed by simulation studies in section three. Section four shows and discusses the results. Finally section five summarizes the paper and brings the conclusions.

**2. Proposed Method.** In this section we briefly discuss political competitions in a parliament and the way we use them as an inspiration source to formulate a new global optimization algorithm.

A parliamentary system, also known as the parliamentarianism is a system of government in which the power to make and execute the laws is held by the parliament. Members of the parliament are elected in general elections. People usually vote in favor of the parties rather than particular persons. Members of a parliament normally belong to political parties. They support their parties in parliamentary votes. Grouping members of a parliament into clusters, results in competitions among parties in trying to gain superiority over other parties. Almost in all democratic countries, political parties form the population of the parliaments.

There are basically two systems in parliamentary elections: the majority election system and the proportional representation system. In the majority election system, only one member of a parliament is elected per constituency. In the proportional representation system several members of the parliament are elected per constituency. Basically every political party presents a list of candidates. Voters then can select the list of their favorite parties. Parties are assigned parliamentary seats proportionally to the number of votes they gain in elections [20].

Political parties, either in or out of the parliament, have members with different levels of capability and power. Those main people of a party try to make good influences over other regular members with less power. They do so to benefit from their supports during chairman elections. Therefore, important members (candidates) of parties compete to achieve the maximum number of followers among regular members. On the other hand, regular members have tendency toward more capable persons and usually vote for those who they believe in and might benefit from. In this mutual and dynamic process regular members with higher capability could replace previous candidates. These competitions are among individuals within parties. Another kind of competition and cooperation is at the level of parties. Political parties compete or cooperate to gain more power than others. Two main goals that parties try to achieve are greater number of seats in the parliament and taking the control of the government. Sometimes parties join or share their resources to gain dominance over other parties to get better opportunities.

Optimization process in our algorithm is started by first creating a population of individuals. These individuals are assumed to be the members of the parliament. In the next step, population is divided into some political groups (parties) and a fixed number of members with highest fitness are selected as group candidates or leaders. After population partitioning, intra-group competition begins. In the intra-group competition phase, each regular member is biased toward all candidates in proportion to their fitnesses. This operation is motivated by the fact that a regular member of a party is usually under the influence of other superior members. This observation is modeled here as the weighted average of distance vectors from a regular member to candidates. This causes the evolutionary process to search for potential points in search space and provides an exploratory mechanism. At the end of intra-party competition, a few candidates with highest fitness are elected as final candidates of each group. They compete with candidates of other groups in the next stage. Both candidates and regular members of a group are important in determining the total power of a group. A weighted linear combination of the mean power of candidates and the mean power of regular members is considered to be the total fitness of a group. Like parliamentary systems of some countries, no voting mechanism is assumed here. In fact the biasing mechanism could somehow be considered as the implicit voting.

Inter-group competition begins just after intra-group competitions end. Political groups within the parliament compete with other groups to impose their own candidates to them. In our method, the role of groups is still preserved after introducing a candidate. Each group who is not further able to compete with others becomes gradually weaker and loses its chance to get the chairman position. Groups with negligible fitnesses gradually lose their power and ultimately collapse. On the other hand, stronger groups become progressively more powerful and consequently earn higher chance to win the competition.

Powerful groups sometimes agree to join and merge into one (at least on some special occasions) to increase their chance to win. This gives the algorithm the chance to search more promising areas for optimal solutions. The tendency of regular members toward group candidates along with affinity of powerful groups to join and also the collapse mechanism drives convergence to a state in which there exists just one group in the parliament. In contrast to what happens in real world, when algorithm converges, regular members have near the same or equal power as the candidate which is now the chairman. A step by step description of the algorithm is shown in Table 1.

TABLE 1. Overall process of the parliamentary optimization algorithm (POA)

Step 1: Initialize the population
Step 2: Partition population into M groups each with L individuals
Step 2.1: Pick $\theta$ highly fitted individuals as candidates of each group
Step 3: Intra-group competition
Step 3.1: Bias regular members toward candidates of each group
Step 3.2: Reassign new candidates
Step 3.3: Compute power of each group
Step 4: Inter-group cooperation
Step 4.1: Pick $\lambda$ most powerful groups and merge them with probability $P_m$
Step 4.2: Remove $\gamma$ weak groups with probability $P_d$
Step 5: If stopping condition is not met go to step 3
Step 6: Report the best candidate as the solution of the optimization problem

**2.1. Population initialization.** A population of N initial solutions is being dispread over the d-dimensional problem space at random positions. Each individual of the population is coded as a d-dimensional continuous vector as in equation 1.

$$P = [p_1, p_2, \dots, p_d], p_i \in \mathbb{R} \quad (1)$$

Each individual could be either a regular member or a candidate of a given group. A fitness function  $f$  is used to calculate the strength of an individual.

**2.2. Population partitioning.** In order to form initial groups, population is partitioned into M divisions. Each group contains L individuals. N, M and L are positive integers and are selected in such a way that N is equal to the multiplication of M and L. Top  $\theta$  candidates with high fitness are then considered as candidates of each group. At this point all groups have the same number of members, but during running the algorithm groups might earn different number of individuals because of merge and collapse mechanisms.

**2.3. Intra-group competition.** Regular members of a group are biased toward candidates after interactions take place between candidates and regular members. Biasing operation is assumed here to be linearly proportional to the weighted average of vectors connecting a member to candidates. Each candidate is weighted to the extent of its fitness value as shown in equation 2.

$$p' = p_0 + \eta \left( \frac{\sum_{i=1}^{\theta} (p_i - p_0) \cdot f(p_i)}{\sum_{i=1}^{\theta} f(p_i)} \right) \quad (2)$$

In the above formula,  $\eta$  is a random number between 0.5 and 2 and allows the algorithm to search in a local search area around candidates. Another alternative mechanism is to use large values of  $\eta$  at first iterations and then gradually reduce it, perhaps by analyzing variance. A regular member is allowed to change only if it takes a higher fitness value. After biasing, regular members might have higher fitness values than candidates. In such cases candidates are reassigned again.

Figure 1 illustrates the biasing operation.  $P_0$  is a regular member and  $P_i$  is a candidate.  $P'$  is the new position of the regular member.

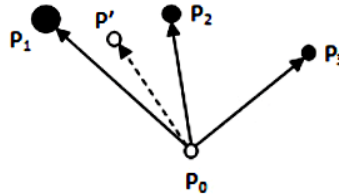


FIGURE 1. Biasing a member of population toward candidates. Influence of each candidate is weighted by its fitness value.

Let  $Q_i = \{Q_{i,1}, Q_{i,2}, \dots, Q_{i,\theta}\}$  and  $R_i = \{R_{i,\theta+1}, R_{i,\theta+2}, \dots, R_{i,L}\}$  be the vectors of candidates and the remaining regular members of the  $i$ -th group respectively. Power of this group is calculated as in equation 3, in which  $m$  and  $n$  are candidate and regular member weighting constants.

$$power^i = \frac{m \cdot Avg(Q_i) + n \cdot Avg(R_i)}{m + n}; m \geq n \quad (3)$$

**2.4. Inter-group cooperation.** Stronger groups sometimes join and merge to one group in order to strengthen their power. To perform merging, a random number is generated and if it is smaller than  $P_m$ ,  $\lambda$  most powerful groups are picked and merged into one. During the course of running the algorithm, weak groups are removed for saving computation power and reducing the function evaluations. Like merging, a random number is generated and if it is smaller than  $P_d$ ,  $\gamma$  groups with minimum powers are eliminated. Figure 2 illustrates the inter-group cooperation between two sample groups. Candidates are reassigned after this operation.

**2.5. Stopping condition.** At the end of the algorithm, a group wins the competitions and its best member (candidate with the maximum fitness) is considered as the solution of the optimization problem. Two stopping conditions are possible: Algorithm terminates if either maximum number of iterations is reached or, during some successive iterations, no significant enhancement in fitness values is observed.

The whole optimization process is shown in flowchart of the Figure 3.

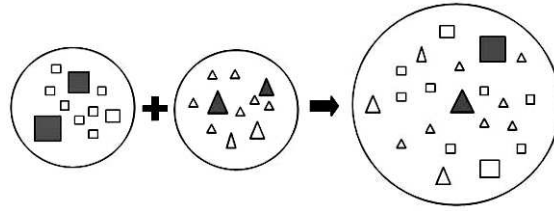


FIGURE 2. Merging two groups into one. Candidates of the new group are reassigned based on their fitnesses.

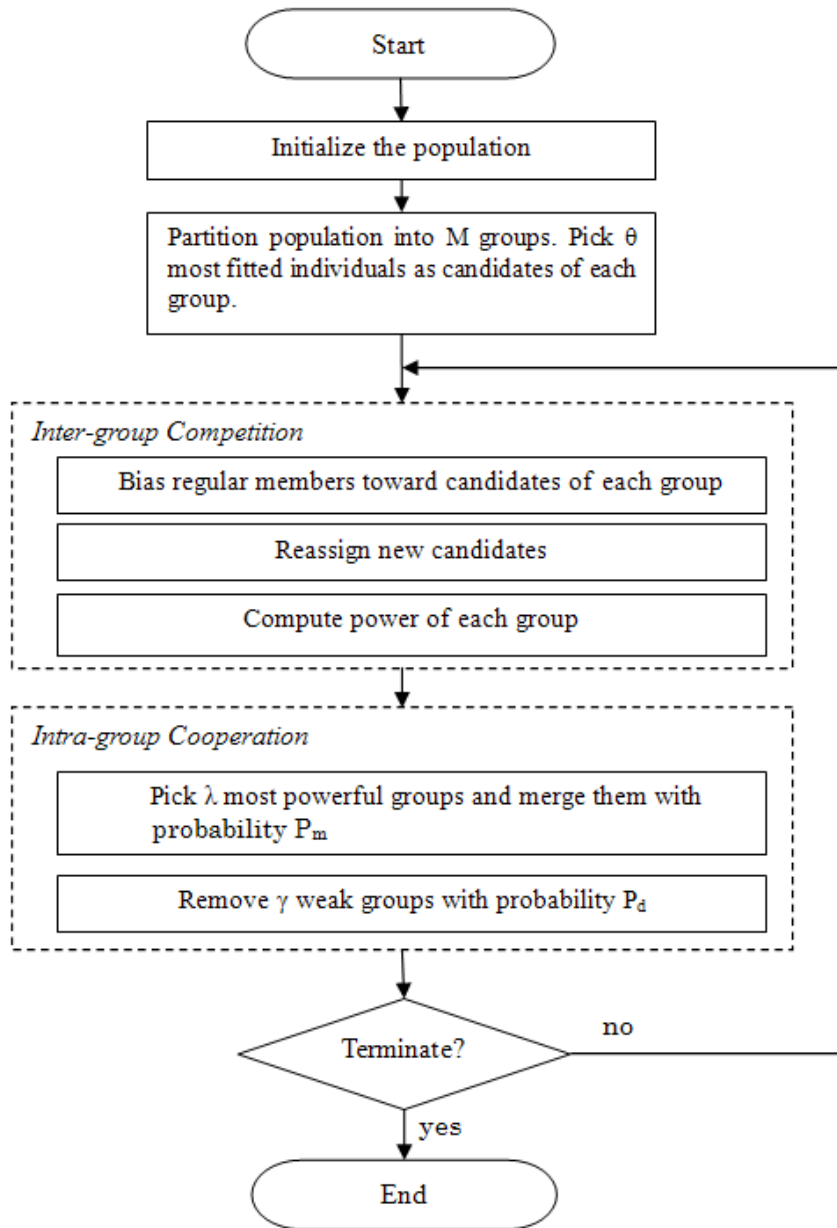


FIGURE 3. Flowchart of the proposed algorithm

**3. Experiments.** In this section we evaluate the POA over six test functions and compare it with the PSO approach. A brief introduction to standard PSO is explained as follows. Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling [11]. In PSO, the potential solutions, called particles, move through the problem space by following the paths of their successful neighbors. Each individual (particle) keeps track of its past best performance (fitness) and that of its neighbors (social view) within a fixed radius and uses this information to determine its next direction and velocity (cognitive view). PSO is initialized with a group of random particles (solutions) and then searches for the optimal points by updating generations. Velocity and position update formulas in PSO are shown in equations 4 and 5.

$$V_i^d(t + 1) = V_i^d(t) + c_1.rand1_i^d(t).(pbest_i^d(t) - X_i^d(t)) + c_2.rand2_i^d(t).(gbest_i^d(t) - X_i^d(t)) \tag{4}$$

$$X_i^d(t + 1) = X_i^d(t) + V_i^d(t + 1) \tag{5}$$

In our experiments,  $c_1$  and  $c_2$  parameters in the above formulas were set to 1.0 and 2.0 respectively. Random numbers rand1 and rand2 were selected ranging from 0 to 1. Symbols d, t and i represent dimensions, iterations and particles indices, respectively.

**3.1. Test functions.** Formulas and sketches of six well known benchmark test functions for numerical optimization that we will use in our experiments are shown in Figure 4. They are called in order Sphere, Rosenbrock, Ackley, Griewank, Non-continuous Rastrigin and Schwefel. This set of functions has special properties making them suitable for evaluating optimization algorithms. They all have a unique global minimum point while some of them have suboptimal local minima. Some of them are non-convex like Rosenbrock and some are convex like sphere. Some have saddle points like Schwefel and Griewank.

In the experiments for all test functions we aim to find the minimum of the fitness value. Initial range, search range, minimum values and the point at which each function takes its minimum value are listed in Table 2. Initial ranges are only used for initializing the first population over the search space in iteration one. All functions were defined in ten dimensions. Our algorithm as stated in the pseudo-code of Table 1 finds the maximum value of functions therefore we use the negative value of the fitness to turn it into a minimization problem.

TABLE 2. Optimum point  $x^*$ , minimum value  $f(x^*)$ , search and initial ranges of test functions

Function Name	$x^*$	$f(x^*)$	Search Range	Initial Range
Sphere	[0,0,...,0]	0	[-100,100]	[-100,50]
Rosenbrock	[1,1,...,1]	0	[-2.048,2.048]	[-2.048,2.048]
Ackley	[0,0,...,0]	0	[-32.76,32.76]	[-32.768,16]
Griewank	[0,0,...,0]	0	[-600,600]	[-600,200]
Non C. Rastrigin	[0,0,...,0]	0	[-5.12,5.12]	[-5.12,2]
Schwefel	[420.96,420.96,...,420.96]	0	[-500,500]	[-500,500]

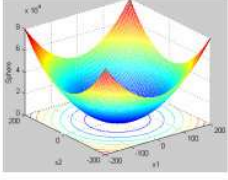
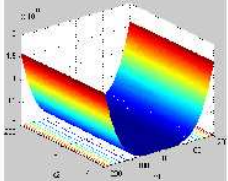
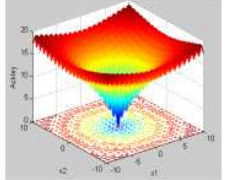
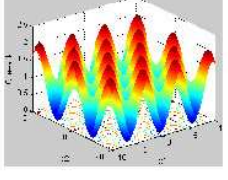
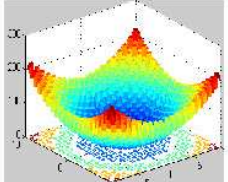
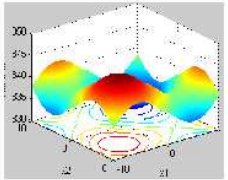
Function	Equation	2D shape
Sphere	$f_1(x) = \sum_{i=1}^D x_i^2$	
Rosenbrock	$f_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	
Ackley	$f_3(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e$	
Griewank	$f_4(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	
NonContinuous Rastrigin	$f_5(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10) \quad \text{for } i = 1, 2, \dots, D$ $y_i = \begin{cases} x_i &  x_i  < 1/2 \\ \text{round}(2x_i) / 2 &  x_i  \geq 1/2 \end{cases}$	
Schwefel	$f_6(x) = 418.9829 * D - \sum_{i=1}^D x_i \sin\left( x_i ^{1/2}\right)$	

FIGURE 4. Formulas and sketches of benchmark test functions used in experiments



**3.2. Setting parameters of POA.** There are a number of parameters that should be set before applying the proposed approach for finding the minimum of a function. Table 3 lists POA parameters and their associated values which were used for all functions. Stopping criteria was the maximum number of iterations which was set to 30000.

TABLE 3. POA parameter values for function minimization

Parameter	Symbol	Value	Parameter	Symbol	Value
Number of groups	N	4	Member weighting constant	n	0.01
Group size	L	5	Candidate weight. constant	m	1
Merge probability	$p_m$	0.01	Groups to be merged	$\lambda$	2
Deletion probability	$p_d$	0.001	Groups to be deleted	$\gamma$	1
Number of candidates	$\theta$	2	Maximum iterations	itr	30000
Bias parameter	$\eta$	0.3			

**4. Results and Discussions.** The performance of POA over test functions listed in Table 2 is shown in Figure 5. The horizontal axis represents the iterations, and the vertical axis represents the fitness values in logarithmic scale. Results of both optimization approaches are averaged over ten different executions. Plots show the fitness of the best and worst individuals and also the mean fitness of the population. Red curves are for the POA and blue ones show the performance of the PSO. Exact numerical values of both algorithms are listed in Table 4.

TABLE 4. Min, mean and max values of the POA and standard PSO over all test functions

		Sphere	Rosenbrock	Ackley
PSO	Min	46.6902 ± 88.9410	47.8131 ± 43.3217	2.5309 ± 1.0014
	Mean	84.1496 ± 9.1433	77.8963 ± 12.6656	2.5309 ± 1.0014
	Max	120.9942 ± 107.6671	105.6791 ± 12.1064	2.5309 ± 1.0014
POA	Min	20.5234 ± 69.8264	51.3701 ± 1.2145e+002	1.8404 ± 0.0167
	Mean	21.0104 ± 78.6304	51.3711 ± 1.2144e+002	2.4587 ± 0.0027
	Max	21.0360 ± 79.1433	51.3711 ± 1.2144e+002	2.7868 ± 0.0016
		Griewank	Non C. Rastrigin	Schwefel
PSO	Min	1.3175 ± 0.0037	23.3492 ± 7.312	4.1505e+3 ± 3.9485e-4
	Mean	1.5092 ± 0.0010	38.9931 ± 0.631	4.1506e+3 ± 6.2824e-5
	Max	1.7316 ± 6.737e-4	50.8212 ± 15.52	4.1506e+3 ± 1.0671e-4
POA	Min	0.8011 ± 0.0576	3.6000 ± 0.800	2.1064e+3 ± 6.6562e-2
	Mean	0.8013 ± 0.0577	6.4520 ± 0.065	2.1064e+3 ± 6.6562e-2
	Max	0.8014 ± 0.0578	9.4000 ± 0.300	2.1064e+3 ± 6.6562e-2

Both approaches were evaluated in detail over six well-known and benchmark test functions. As results show, our proposed approach outperforms the standard PSO approach over all test optimization functions. It worked better over both convex functions and

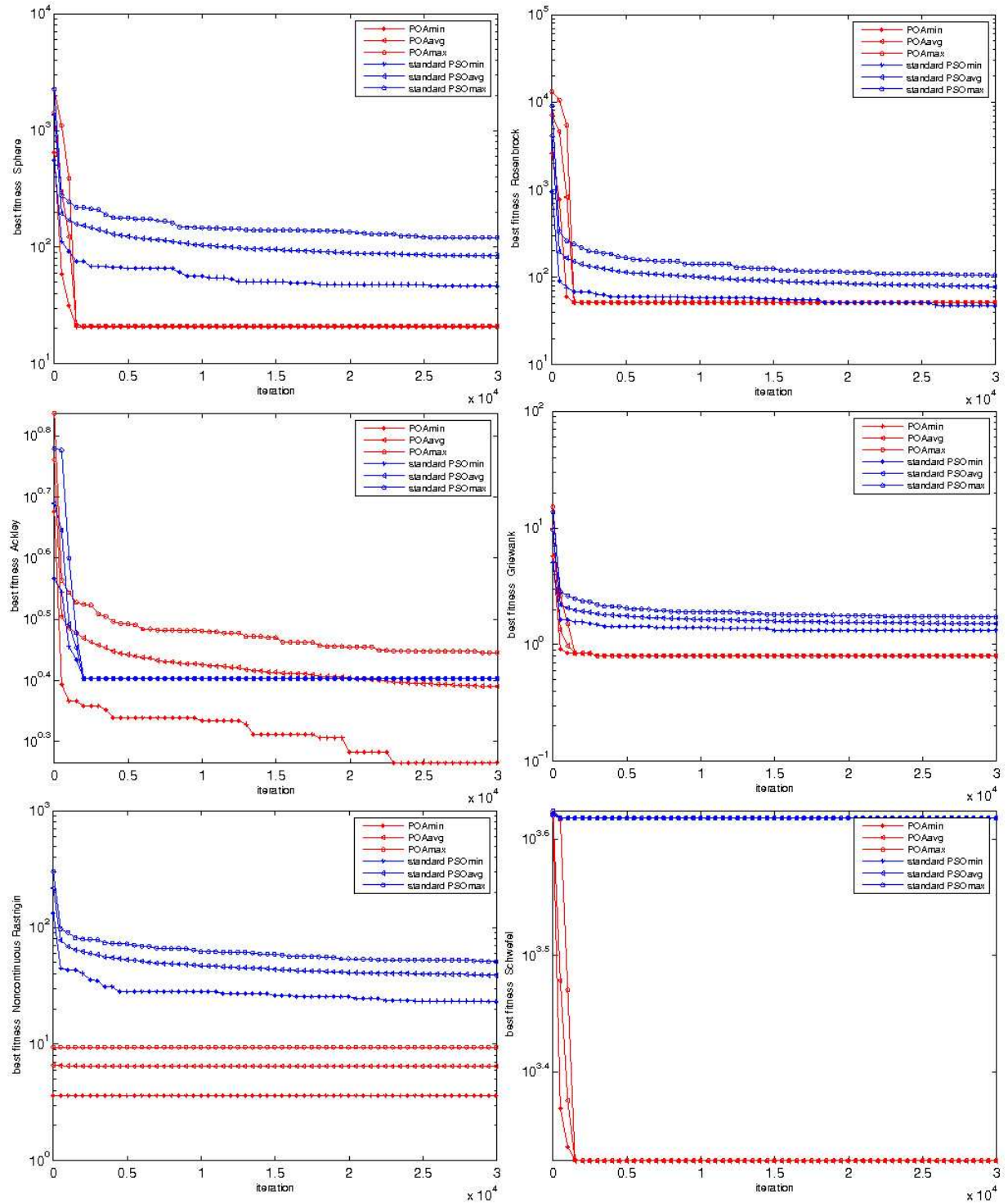


FIGURE 5. The experimental results of POA over test functions

functions with many local minima. This better performance can be described to PSOs possibility of premature convergence, without reaching a local or global minimum and POAs ability to escape from local minima and its exploratory power.

By analyzing the results of the proposed algorithm on 10 dimensional problems, it could be concluded that POA performs the best for the Schwefel function compared with the standard PSO. In all cases, POA converged faster than PSO to better solutions in the final generation. Over all functions except Ackley, nearly in generation 1500, even curves of the max (worst) individual of the POA was below the min (best) of the PSO approach. POA worked worse than PSO over Ackley function in average, but it succeeded to find better solutions in the last generation. It could be proved from figure 5 that POA has a higher explanatory power than the standard PSO approach.

**5. Conclusions.** Biology inspired computation considers different computational approaches that are based on the models of biological systems. Recent approaches in computational intelligence are shifted toward inspiration from human behaviors in different aspects of social life. An important application area to evaluate such approaches is numerical optimization. In this work, we introduced a new global optimization algorithm which is inspired from competitions among political parties in trying to take the control of a parliament. Several numerical simulations were carried out to investigate the convergence of POA over six benchmark optimization problems. Our experimental results indicate that POA outperforms PSO on finding the global best solutions.

As shown by simulations, our proposed algorithm captures important spirits of parliamentary political interactions fairly well. It is also capable of finding desired minima very fast in comparison with the other stochastic search methods. As an optimization algorithm, it has the additional desirable properties of capability to deal with complex and non-differentiable objective functions as well as escaping from local minima.

Some future directions are taking into account the more sophisticated and complex political behaviors and comparing the derived methods with the advanced PSO and other optimization methods. Applying this method for solving real world applications in engineering and life sciences is also another interesting future work. Large set of parameters is a weakness of the algorithm and should be tackled in future versions.

## REFERENCES

- [1] J. H. Holland, *Adoption in Natural and Artificial Systems*, University of Michigan, Ann Arbor, 1975.
- [2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, 1989.
- [3] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
- [4] K. W. Kim, M. Gen and M. H. Kim, Adaptive genetic algorithms for multi-resource constrained project scheduling problem with multiple modes, *International Journal of Innovative Computing, Information and Control*, vol.2, no.1, pp.41-49, 2006.
- [5] L. DeCastro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, 2001.
- [6] V. Cutello, and G. Nicosia, An immunological approach to combinatorial optimization problems, *Lecture Notes in Computer Science*, vol.2527, pp.361-370, 2002.
- [7] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, vols. 1 and 2, Cambridge: MIT, 1986.
- [8] U. Wild, and A. Renn, Molecular computing - a review .1. data and image storage, *Journal of Molecular Electronics*, vol.7, no.1, pp.1-20, 1991.

- [9] L. Huang, and N. Wang, An optimization algorithm inspired by membrane computing, *Advances in Natural Computation*, Pt 2, vol.4222, pp.49-52, 2006.
- [10] H. Q. Qu, H. Zhu and C. Peng, New algorithms for some NP-optimization problems by DNA computing, *Progress In Natural Science*, vol.12, no.6, pp.459-462, 2002.
- [11] J. Kennedy, and R. Eberhart, Particle swarm optimization, *Proc. of the IEEE International Conference on Neural Networks*, pp.601-610, 1995.
- [12] J. F. Schutte, and A. A. Groenwold, A study of global optimization using particle swarms, *Journal of Global Optimization*, vol.31, no.1, pp.93-108, 2001.
- [13] Y. Shi, and R. Eberhart, Empirical study of particle swarm optimization, *Proc. of the Congress on Evolutionary Computation*, pp.1945-1950, 1999.
- [14] Z. Cui, G. Sun and J. Zeng, A fast particle swarm optimization, *International Journal of Innovative Computing, Information and Control*, vol.2, no.6, pp.1365-1380, 2006.
- [15] E. Bonabeau, M. Dorigo and G. Theraulaz, Inspiration for optimization from social insect behavior, *Nature*, vol.406, no.6791, pp.39-42, 2000.
- [16] M. Dorigo, V. Maniezzo and A. Colorni, The ant system: optimization by a colony of cooperating agents, *IEEE Trans. on Systems Man and Cybernetics Part B-Cybernetics*, vol.26, no.1, pp.1-13, 1996.
- [17] S. C. Chu, and P. W. Tsai, Computational intelligence based on the behavior of cats, *International Journal of Innovative Computing, Information and Control*, vol.3, no.1, pp.163-173, 2007.
- [18] T. Ray, and K. M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behavior, *IEEE Trans. on Evolutionary Computation*, vol.7, no.4, pp.386-396, 2003.
- [19] K. Sharples, Socio-cognitive engineering: a methodology for the design of human-centered technology, *European Journal of Operation Research*, vol.136, no.2, pp.310-323, 2002.
- [20] A. Shourie, *The Parliamentary System*. Rupa & Co, USA, 2007.