IJST, Transactions of Civil Engineering, Vol. 37, No. C+, pp 491-501
Printed in The Islamic Republic of Iran, 2013

EMPSACO: AN IMPROVED HYBRID OPTIMIZATION ALGORITHM BASED ON PARTICLE SWARM, ANT COLONY AND ELITIST MUTATION ALGORITHMS*

A. KHASHEI- SIUKI[1,**] I. TADAYONI NAVAEI[2], B. GHAHRAMAN[3] AND M. KOUCHAKZADEH[4]

[1]Dept. of Water Engineering, University of Birjand, I. R. of Iran
Email: Abbaskhashei@birjand.ac.ir
[2] Dept. of Mechanical Engineering, Islamic Azad University of Mashhad, I. R. of Iran
[3]Dept. of Irrigating, Ferdowsi University of Mashhad, Mashhad, I. R. of Iran
[4]Dept. of Irrigating and Drainage Eng., Tarbiat Modares University, Tehran, I. R. of Iran

Abstract– This research presents an efficient and reliable swarm intelligence-based approach, ant colony optimization and elitist-mutated particle swarm optimization. Methods of particle swarm optimization (PSO) and ant colony optimization (ACO) and elitist mutation particle swarm optimization (EMPSO) are co-operative, population-based global search swarm intelligence metaheuristics. PSO is inspired by social behavior of bird flocking or fish schooling, while ACO imitates foraging behavior of real life ants and Elitist mutation taken from genetic mutation from genetic algorithm techniques. In this study, we explore a simple approach to improve the performance of the PSO method for optimization of multimodal continuous functions. The proposed EMPSACO algorithm is tested on several test functions from the usual literature and compared with PSO, PSACO and GA (Genetic Algorithm). Results showed that the effectiveness and efficiency of the proposed EMPSACO method had suitable accuracy to optimize multimodal functions.

Keywords– Particle swarm optimization, ant colony, elitist mutation, metaheuristics, EMPSACO

1. INTRODUCTION

In spite of the development of many conventional techniques for optimization, each of these techniques has its own limitations. Generally, optimization problems are divided into two general categories: A) problems by which objective functions are differentiable and are known as classical techniques. B) Problems that are non-derivative or discontinuous objective functions. Classical methods for solving these functions cannot be used. Hence numerical methods have been developed for this category.

So methods and models have limitations for optimization of decision variables in the objective function. These restrictions include the multiple variables and constraints and nonlinear and nonconvex or discontinuous objective function.

To overcome those limitations, recently metaheuristic techniques have been used for optimization. By using these techniques, the given problem can be represented more realistically. These also provide ease in handling the nonlinear and nonconvex relationships in the formulated model. Genetic algorithms (GA) [1] and particle swarm optimization (PSO) [2] are some of the techniques in this category. These evolutionary algorithms search from a population of points, so there is a greater possibility to cover the whole search space and reach the global optimum. Particle swarm optimization can be and has been used across a wide range of applications. Areas where PSOs have shown particular promise include multimodal

*Received by the editors April 20, 2011; Accepted October 21, 2013.
**Corresponding author

problems and problems for which there is no specialized method available or all specialized methods give unsatisfactory results. The standard PSO algorithm due to be trapped into local optimum solutions can't provide accurate and certain answers.

Particle swarm optimization (PSO) was originally designed and introduced by Eberhart and Kennedy ([2]; [3]; [4]). The PSO is a population based search algorithm based on the simulation of the social behavior of birds, bees or a school of fishes. This algorithm originally intends to graphically simulate the graceful and unpredictable choreography of a bird folk.

Each individual within the swarm is represented by a vector in multidimensional search space. This vector also has one assigned vector which determines the next movement of the particle and is called the velocity vector. PSO is an efficient tool for optimization and search problems. Researchers improved it with different methods However, it is easy to be trapped into local optima due to its information sharing mechanism.

Parsopoulos and Vrahatis [5] studied the behavior of the PSO when Gaussian distributed random noise was added to the fitness function and random rotations of the search space were performed. Experimental results indicated that the PSO remained effective in the presence of noise, and, in some cases, noise even helped the PSO avoid being trapped in local optima.

In the research the PSO was compared to a noise-resistant variant where the main PSO loop was modified so that multiple evaluations of the same candidate solution are aggregated to better assess the actual fitness of this particular solution. The comparison considered several numerical problems with added noise, as well as unsupervised learning of obstacle avoidance using one or more robots. The noise-resistant PSO showed considerably better performance than the original [6].

Angeline [7] produced one of the first intentionally hybridized particle swarms. In his model, selection was applied to the particle population; "good" particles were reproduced with mutation, and "bad" particles were eliminated. Angeline's results showed that PSO could benefit from this modification.

Miranda and Fonseca [8] borrowed an idea from evolution strategies. In that paradigm, points are perturbed by the addition of random values distributed around a mean of zero; the variance of the distribution is evolved along with function parameters. Those researchers used Gaussian random values to perturb W, C_1, and C_2, (W is the particle inertia, C_1 and C_2 are the cognitive and social scaling parameters) as well as the position of the neighborhood best but not the individual best using selection to adapt the variance. The evolutionary self-adapting particle swarm optimization method, a hybrid of PSO and evolutionary methods, has shown excellent performance in comparison to some standard particle swarm methods. Miranda has used it for the manufacture of optical filters as well as in the optimization of power systems.

In research it took a different tack, embedding velocity information in an evolutionary algorithm. They replaced Cauchy mutation with a version of PSO velocity in a fast evolutionary Programming (FEP) algorithm, to give the FEP population direction. Their published results indicate that the approach is very successful in a range of functions; the new algorithm found global optima in tens of iterations, compared to thousands for the FEP versions tested [9].

Robinson et al., [10] trying to optimize a profiled corrugated horn antenna, noted that a GA improved faster early in the run, and PSO improved later. As a consequence of this observation, they hybridized the two algorithms by switching from one to the other after several hundred iterations. They found the best horn by going from PSO to GA (PSO-GA) and noted that the particle swarm by itself outperformed both the GA by itself and the GA-PSO hybrid, though the PSO-GA hybrid performed best of all. It appears from their result that PSO most effectively explores the search space for the best region, while GA is effective in finding the best point once the population has converged on a single region; this is consistent with other findings.

In the research Kaveh and Malakoutirad [11] presented an evolutionary algorithm based on the hybrid genetic algorithm (GA) and particle swarm optimization (PSO), denoted by HGAPSO, is developed in order to solve force method-based simultaneous analysis and design problems for frame structures. Suitability of the HGAPSO algorithm is compared to both GA and PSO for all the design examples, demonstrating its efficiency and superiority, especially for frames with a larger number of redundant forces.

In another research by Kaveh and Masoudi [12] an efficient algorithm was developed for the formation of null basis of triangular and rectangular plate bending finite element models, corresponding to highly sparse flexibility matrices. This is achieved by applying a modified ant colony system. An integer linear programming formulation is also presented to evaluate the quality of the results obtained by the proposed ant colony system algorithm. The efficiency of the present algorithm is illustrated through some examples.

Clerc's recent experiments have shown that adaptation of the constriction factor, population size, and number of neighbors can produce improved results. His studies found that best performance was obtained when all three of these factors are adapted during the course of the run. Clerc used three rules: (a) Suicide and generation: a particle kills itself when it is the worst in its neighborhood and generates a new copy of itself when it is the best; (b) Modifying the coefficient: good local improvement caused an increase in the constriction coefficient, while poor improvement caused its decrease; (c) Change in neighborhood: the locally best particle could reduce the number of its neighbors, while poorly performing particles could increase theirs. Adaptive changes were not made on every iteration, but only occasionally [13].

This paper proposes a combination of improved particle swarm optimization hybridized with an ant colony approach and elitist mutation particle swarm optimization (EMPSO) called EMPSACO (elitist mutation particle swarm ant colony optimization), for optimization of multimodal continuous functions. The proposed method applies PSO for global optimization and the idea of the ant colony approach to update the positions of particles to rapidly attain the feasible solution space. Although this method may easily to be trapped into local optima, elitist mutation approach releases it.

The implementation of EMPSACO algorithm consists of three stages. In the first stage, it applies PSO, while ACO is implemented in the second stage. ACO works as a local search, wherein ants apply pheromone-guided mechanism to update the positions founded by the particles in the earlier stage. The implementation of ACO in the second stage of EMPSACO is based on the studies by Angeline [7] which showed that (1) PSO discovers reasonable quality solutions much faster than other evolutionary algorithms, (2) If the swarm is going to be in equilibrium, the evolutionary process will be stagnated as time goes on. Thus, PSO does not possess the ability to improve upon the quality of the solutions as the number of generations is increased. 3) The role of elitist mutation in stage three of EMPSACO is based on studies (Nagesh Kumar and Janga Reddy [14]) which demonstrate that EMPSO consistently performs better than the standard PSO and genetic algorithm techniques. The purposes of this research are 1) improved PSO algorithm 2) hybridized ACO algorithm and elitist mutation from genetic algorithm 3) to compare different methods and find better quality solutions with less computational time and 4) to compare PSO hybrid algorithm with GA method

2. MATERIALS AND METHODS

In this study, codes of PSO were written in Fortran 90 software. The proposed EMPSACO method is tested on several widely used benchmark multimodal continuous functions [15]. Numerical results are compared with the some other hybrid PSO methods available in the usual literature and GA.

a) Particle swarm optimization

The term "swarm intelligence" is used to describe algorithms and distributed problem solvers, which was inspired by the collective behavior of insect colonies and other animal societies. Under this prism, PSO is a swarm intelligence method for solving optimization problems. Particle swarm optimization is a population-based heuristic search technique, inspired by social behavior of bird flocking. PSO shares many similarities with evolutionary computation techniques such as GA. PSOs are initialized with a population of random solutions and search for optima by updating generations.

However, in contrast to methods like GAs, in basic PSO, no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead, PSO relies on the exchange of information between the individual particles of the population swarm. In effect, each particle adjusts its trajectory towards its own previous best position and towards the current best positions attained by any other member in its neighborhood [5].

b) PSO algorithm

In PSO, candidate solutions of a population, called particles, coexist and evolve simultaneously based on knowledge sharing with neighboring particles. While flying through the problem search space, each particle generates a solution using directed velocity vector. Each particle modifies its velocity to find a better solution (position) by applying its own flying experience (i.e. memory having best position found in the earlier flights) and experience of neighboring particles (i.e. best-found solution of the population). Particles update their velocities and positions as shown below:

$$v^{i}_{t+1} = w_t\, v^{i}_{t} + c_1 r_1 (p^{i}_{t} - x^{i}_{t}) + c_2 r_2 (p^{g}_{t} - x^{i}_{t}), \tag{1}$$

$$x^{i}_{t+1} = x^{i}_{t} + v^{i}_{t+1} \tag{2}$$

where x^{i}_{t} represents the current position of particle i in solution space and subscript t indicates an iteration count; p^{i}_{t} is the best-found position of particle i up to iteration count t and represents the cognitive contribution to the search velocity v^{i}_{t}. Each component of v^{i}_{t} can be clamped to the range $[-v_{max}, v_{max}]$ to control excessive roaming of particles outside the search space; p^{g}_{t} is the global best-found position among all particles in the swarm up to iteration count t and forms the social contribution to the velocity vector; r_1 and r_2 are random numbers uniformly distributed in the interval (0, 1), while c_1 and c_2 are the cognitive and social scaling parameters, respectively; w_t is the particle inertia, which is reduced dynamically to decrease the search area in a gradual fashion.

Particle i flies toward a new position according to Eqs. (1) and (2). In this way, all particles P of the swarm find their new positions and apply these new positions to update their individual best p^{i}_{t} points and global best p^{g}_{t} of the swarm. This process is a reiteration until iteration count $t = t_{max}$ (a user-defined stopping criterion is reached).

c) Ant colony optimization (ACO)

ACO is a multi-agent approach that simulates the foraging behavior of ants for solving difficult combinatorial optimization problems, such as the traveling salesman problem and the quadratic assignment problem [16]. Ants are social insects whose behaviors are directed more toward the survival of the colony as a whole than that of a single individual of the colony. An important and interesting behavior of an ant colony is its indirect co-operative foraging process. While walking from food sources to the nest and vice versa, ants deposit a substance, called pheromone on the ground and form a pheromone trail.

This section describes the implementation of the proposed improvement in particle swarm optimization using an ant colony approach. The proposed method, called PSACO (particle swarm ant colony

optimization) is based on the common characteristics of both PSO and ACO algorithms, like survival as a swarm (colony) by coexistence and cooperation, individual contribution to food searching by a particle (an ant) by sharing information locally and globally in the swarm (colony) between particles (ants), etc. The implementation of PSACO algorithm consists of two stages. In the first stage, it applies PSO, while ACO is implemented in the second stage. ACO works as a local search, wherein ants apply pheromone-guided mechanism to refine the positions found by particles in the PSO stage. In PSACO, a simple pheromone-guided mechanism of ACO is proposed to apply as local search. The proposed ACO algorithm handles P ants equal to the number of particles in PSO. Each ant i generates a solution z^i_t around p^g_t, the global best-funded position among all particles in the swarm up to the iteration count it as

$$z^i_t = N(p^g_t; \sigma) \tag{3}$$

In Eq. (3), we generate components of solution vector z^i_t, which satisfy Gaussian distributions with mean p^g_t and standard deviation σ, where initially at t = 1 value of σ = 1 and is updated at the end of each iteration as $\sigma = \sigma \cdot d$, where d is a parameter in (0.25, 0.997) and if $\sigma < \sigma_{min}$ then $\sigma = \sigma_{min}$, where, σ_{min} is a parameter in (10^{-2}, 10^{-4}). Compute objective function value z^i_t and replace position x^i_t the current position of particle i in the swarm if its objective function was more than particle i in PSO[15].

This simple pheromone-guided mechanism considers the highest density of trails (single pheromone spot) at the global best solution p^g_t of the swarm at any iteration t + 1 in each stage of ACO implementation and all ants P searches for better solutions in the neighborhood of the global best solution. In the beginning of the search process, ants explore the larger search area in the neighborhood of p^g_t due to the high value of standard deviation σ and intensify the search around

p^g_t as the algorithm progresses. Thus, ACO helps PSO process not only to efficiently perform global exploration for rapidly attaining the feasible solution space but also to effectively reach optimal or near optimal solution.

d) Elitist mutation

To improve the performance of the code PSACO, in this study, a strategic mechanism called *elitist mutation* is incorporated ino the algorithm. This elitist process replaces the worst particle solutions by the best solution among the swarm after performing mutation mechanism on the best particle[14].

In this method three process of random perturbation tries to improve the solution by maintaining diversity in the population, and explores the new regions in the whole search space. This strategy replaces the position vectors of a predefined number of least ranked particles in the swarm with mutated positions of the global best particle in each iteration. In the EMPSACO methodology, the elitist-mutation step is computed as follows. First the fitness function of particles is sorted in ascending order and the index number for the respective particles is obtained, then elitist-mutation is performed on worst fitness particles in the swarm. Let NM_{max} be the number of particles to be elitist-mutated; p_{em}: probability of mutation; ASF: index of sorted population; rand=uniformly distributed random number $U(0,1)$; rand$n$=Gaussian random number $N(0,1)$; and $VR(d)$=range of decision variable d

For i=1 to NM_{max}
l=ASF(i)
For d=1 to dim
if (rand<p_{em})
$X(l)(d)=P(g)(d)+0.1*\ VR(d)*$rand$n$
else
$X\_l\_(d)=P(g)(d)$

If the mutated value exceeds the bounds, then it is limited to the upper or lower bound. During this elitist-mutation step, the velocity vector of the particle is unchanged.

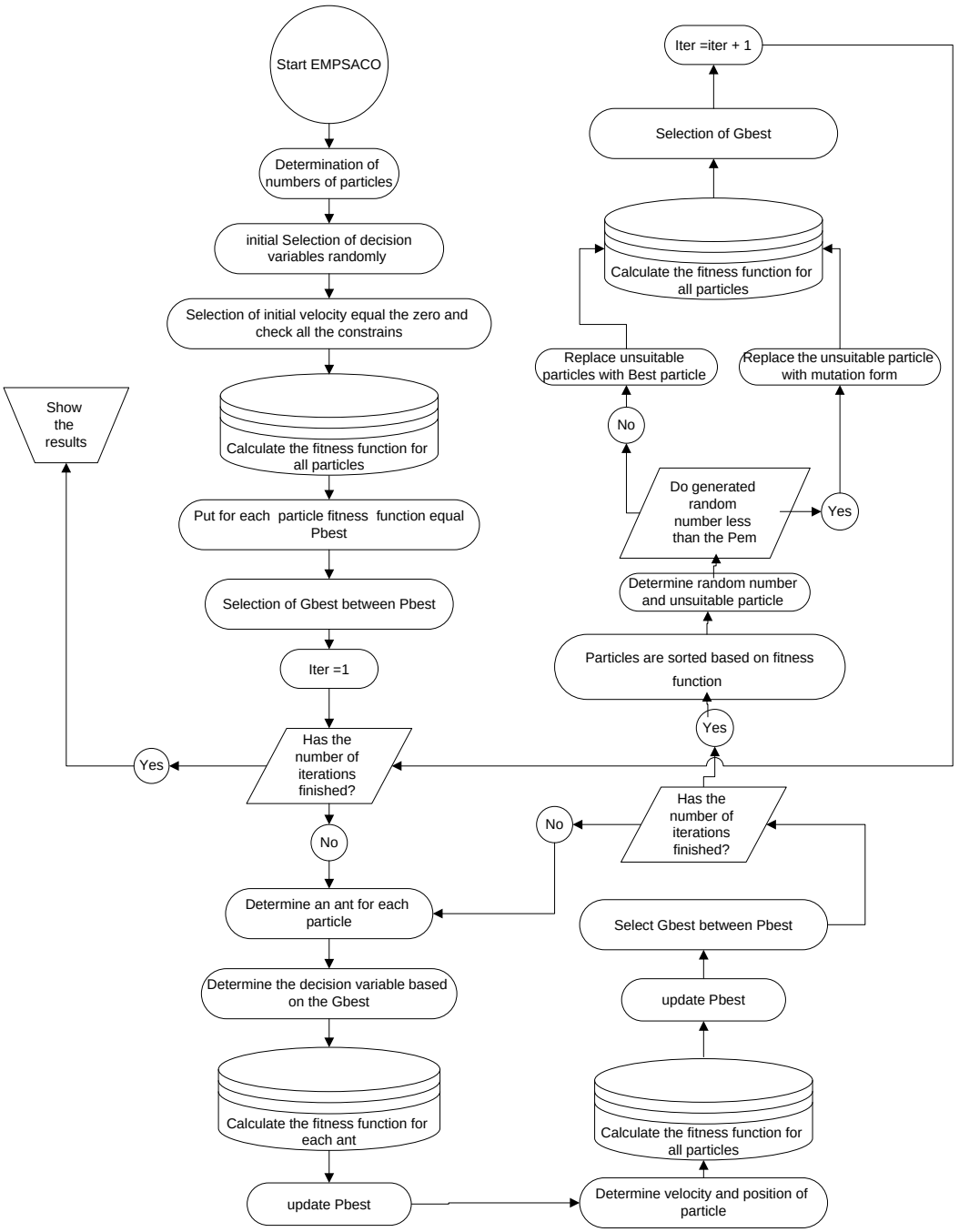

Fig. 1. The flow chart of different stage of optimization results by EMPSACO algorithm

3. RESULTS AND DISCUSSION

The test functions considered in this study are several well-known benchmark multimodal problems. Therefore, there is a better chance to prove the efficiency of the optimization methods. The details of test functions considered are given in Table 1. To detect optimal parameters, test function was tested by trial and error. PSO algorithm parameter settings used in all the simulations are given as: number of particles, P = 150; cognitive and social scaling parameters, $c_1 = 2$, $c_2 = 2$; maximum and minimum values of inertia

weights, w_{max} = 0.9, w_{min} = 0.4; ([15]; [17]). To compare the performance of the proposed EMPSACO method with the performance of some other existing hybrid PSO methods, several sets of experiments were carried out and the results are given in Table 3 and Figs. 2-8.

To compare the performance of EMPSACO with PSO and PSACO Table 2 reports the produced results of algorithms after 120 function evaluations in terms of mean, maximum, minimum and standard division value of the objective function. It can be observed that the results obtained by each EMPSACO, PSACO and PSO are much closer to the theoretical optima that agree with the results [15]. The proposed EMPSACO method has access to all the methods in terms of the searching quality and derivation of results.

All the test functions are multimodal functions and F1 and F2 are bound and no bound functions which are given in Table 1. Results of seven functions showed that the PSO algorithm has less accuracy than the other algorithms. The proposed EMPSACO method in all tasks needs less iteration to reach objective function. Especially for multimodal functions, algorithms with mutation perform obviously better than PSO. The start points of algorithms in functions of Bohachevsky, Rastrigin, Goldstein–Price, F1 and F2 were the same but for Rosenbrock and Shubert the functions were not the same (Figs. 2-8).

Table 1. Details of test functions, domains and function value

Functions	Variable bounds	Objective	Number of local optimal	Test functions	Optimal value (x1, x2)	Function value
F1(x1,x2)	-3 ≤ x1 ≤ 12.1 4.1≤x2≤5.8	max	6	F(x1,x2)=21.5+x1sin(4πx1)+x2sin(20πx2)	(11.6314, 5.724824)	38.85029446
F2(x1,x2)	G1(x1,x2)=(100-(x1-5)2-(x2-5)2 G2(x1,x2)=x1-6)2+(x2-5)2-82.81	min	-	F(x1,x2)=(X1-10)3+(X2-2)3	(14.095, 0.84296)	-6961. 81388
Goldstein–Price (GP) (2 variables)	-2 ≤ x1, x2 ≤ 2	min	4	$GP(x_1,x_2)=(1+(x_1+x2+1)^2(19-14(x_1+x_2)+3(x_1^2+x_2^2)+6x_1x_2))(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+12x_1^2+48x_2-36x_1x_2+27x_2^2))$	(0,-1)	3
Bohachevsky (2 variables)	-100 ≤ x1 ≤ 100, -100 < x2 < 100	min	Several local	$B2(x_1,x_2)= x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+o.7$	(0, 0),	0
Rastrigin (RA) (2 variables)	-1 ≤ x1, x2 ≤ 1;	min	50	$RA(x_1, x_2) =x^2_1\ x^2_2- \cos(18x_1) - \cos(18x_2)$;	(0, 0),	-2
Shubert (SH) (2 variables)	-10 ≤ x1, x2 ≤ 10	min	760	SH(x1, x2)= $(\sum_{i=1}^{5} i\cos((i+1)x1+i))(\sum_{i=1}^{5} i\cos(i$	18 global minima	-186.7309
Rosenbrock (RS2)	-5 ≤ xj ≤ 10, j = 1,. . . ,n;	min	several local minima (exact number of local minima unspecified in the usual literature)	$RS_n(x)= \sum_{j=1}^{n-1}\{100(x_j^2-x_{j+1})^2+(x_j-1)^2\}$	x = (1, . . . , 1)	0

Results in Fig. 1 show that elitist mutation gives the fastest convergence rate on all test functions. This report corresponds with Wang et al's [18] results.

The performance of EMPSACO algorithm has been further compared with Genetic Algorithm GA [19] Results have been shown in Table 2. It was tested with four functions (Bohachevsky, Rastrigin, Goldstein–Price, Shubert) with fixed iteration. Results show that the PSO hybrid algorithm has better accuracy than the GA algorithm (Figs. 2-8). It can be observed that EMPSACO has obtained solutions with smallest objective function. It can be anticipated that EMPSACO approach remains quite competitive as compared to the other existing methods.

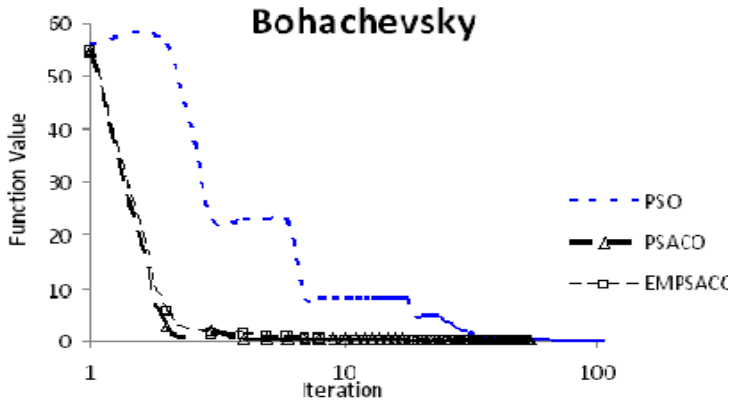

Fig. 2. Comparing convergence of three algorithms for Bohachevsky functions

Fig. 3. Comparing convergence of three algorithms for Goldstein-price functions

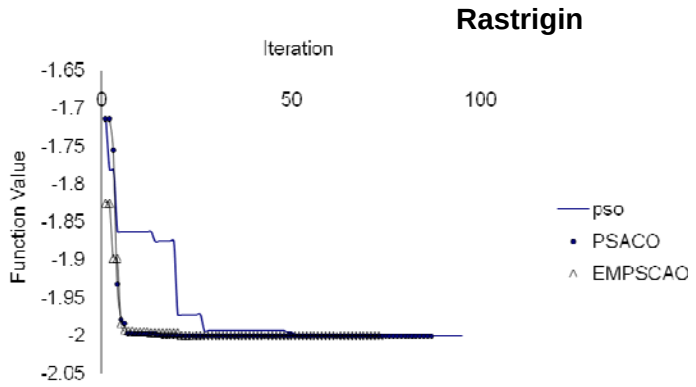

Fig. 4. Comparing convergence of three algorithms for Rastrigin functions

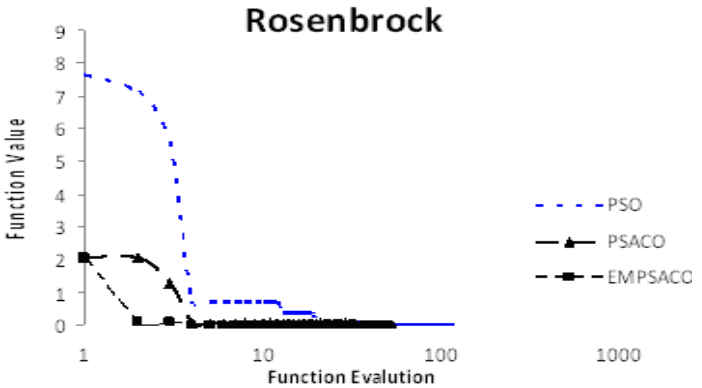

Fig. 5. Comparing convergence of three algorithms for Rosenbrock functions

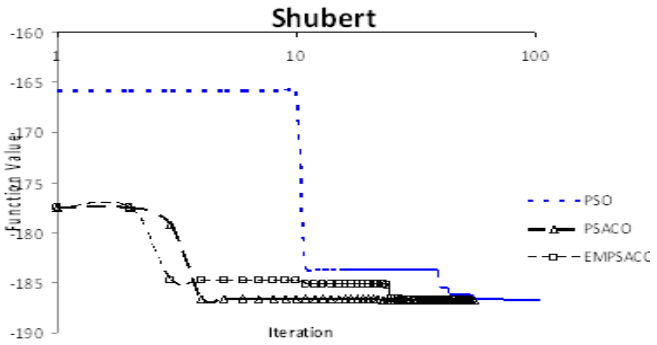

Fig. 6. Comparing convergence of three algorithms for Shubert functions

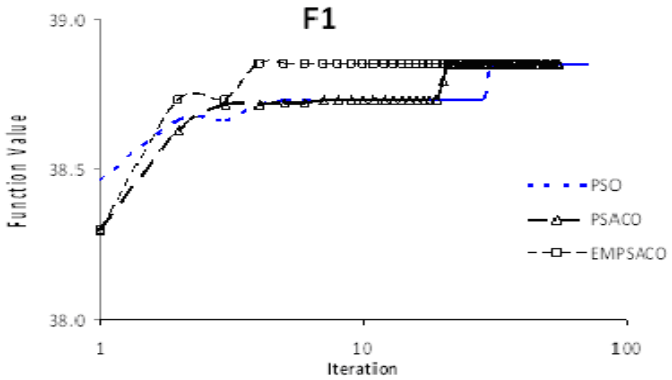

Fig. 7. Comparing convergence of three algorithms for F1 functions

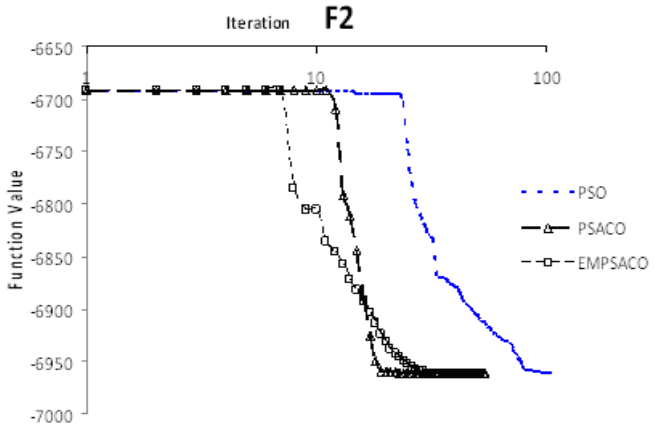

Fig. 8. Comparing convergence of three algorithms in F2 function

Table 2. Fixed iteration experiments results with GA algorithm

Test function	PSO	PSACO	EMPSACO	GA (Liu.et al.,2005)	Optimal value
Goldstein–Price	3.00000033	3.00000044	3.00000005	3.1471	3
Bohachevsky	0.00000012	0.00000011	0.0000001	0.00003	0
Shubert	-186.7309038	-186.7309051	-186.7309061	-182.1840	-186.7309
Rastrigin	-1.9999999	-1.9999998	-1.99999998	-1.9645	-2

Table 3. Statistical parameters of test functions results

	Bohachevsky					F2				
	max	min	mean	Standard deviation	iteration	max	min	mean	Standard deviation	iteration
PSO	55.84899634	0.00000012	3.000023009	8.327757651	119	-6691.82844	-6961.751516	-6900.799741	94.48265298	120
PSACO	54.75567859	0.00000011	1.136750831	7.376295193	54	-6691.82844	-6961.778476	-6891.421259	112.4247881	54
EMPSACO	54.75567859	0.0000001	1.235911688	7.607940185	52	-6691.82844	-6961.785429	-6891.379876	97.89240639	46
	Rastrigin					Rosenbrock				
	max	min	mean	Standard deviation	iteration	Max	min	mean	Standard deviation	iteration
PSO	-1.7137933	-1.9999999	-1.966211304	0.06157743	95	7.6531286	0.0000098	0.267746327	1.084920614	118
PSACO	-1.7137933	-1.9999998	-1.988960715	0.050768534	86	2.07781998	0.00000246	0.122137531	0.427083272	53
EMPSACO	-1.824699	-1.99999998	-1.991034112	0.032780568	73	2.07781998	0.00000616	0.092513036	0.397172256	27
	F1					Goldstein–Price				
	max	min	mean	Standard deviation	iteration	Max	min	mean	Standard deviation	iteration
PSO	38.850294475	38.461835780	38.794851692	0.072257819	71	4.48975219	3.00000033	3.117668605	0.300485809	93
PSACO	38.850294473	38.295268260	38.800687272	0.089952777	58	4.48975219	3.00000044	3.081716853	0.328171167	80
EMPSACO	38.850294478	38.295268260	38.831916150	0.086398289	44	4.48975219	3.00000005	3.086181277	0.335403273	38
	Shubert									
	max	Min	mean	Standard deviation	iteration					
PSO	-165.8556017	-186.7309038	-184.1465604	5.669549108	120					
PSACO	-177.5246005	-186.7309051	-186.392402	1.615779495	84					
EMPSACO	-177.5246005	-186.7309061	-185.7619514	1.7654589	59					

4. CONCLUSION

This paper investigates three operators that are based on the Particle Swarm Optimization algorithm (PSO). In this research a proposed technique of elitist mutation was added to PSACO [15]. By introducing mutation, PSO greatly improves its global search capability without losing its fast convergence property. Though different mutation operators give a different performance on different test problems, the elitist mutation algorithm shows a balanced performance on all test problems and it gives the best performance on some problems. Results showed that the PSO hybrid algorithm has better accuracy than the GA algorithm. It can be observed that EMPSACO obtains solutions with smallest objective function. Hence, the integration of elitist mutation operators is a promising technique for improving the performance of PSO.

REFERENCES

1. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wiley, Reading, N.Y.
2. Eberhart, R. & Kennedy, J. (1995). A new optimizer using particles swarm theory. *Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center*, Piscataway, NJ, pp. 39-43.
3. Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center*, Piscataway, NJ, IV, pp. 1942- 1948,.
4. Kennedy, J. & Eberhart, R. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers, Inc., San Francisco, CA,.
5. Parsopoulos, K. E. & Vrahatis, M. N. (2001). *Particle swarm optimizer in noisy and continuously changing environments*. InM. H. Hamza (Ed.), Artificial intelligence and soft computing. Anaheim: IASTED/ACTA. pp. 289–294

6. Pugh, J., Martinoli, A. & Zhang, Y. (2005). Particle swarm optimization for unsupervised robotic learning. *In Proceedings of IEEE Swarm Intelligence Symposium (SIS)*, pp. 92–99), Piscataway: IEEE.
7. Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), *Proceedings of Evolutionary Programming VII*, pp. 601–610, Berlin: Springer.
8. Miranda, V. & Fonseca, N. (2002). New evolutionary particle swarm algorithm (EPSO) applied to voltage/ VAR control. *In Proceedings of the 14th power systems computation conference (PSCC)*, Session 21, Paper 5, pp. 1–6, Seville, Spain.
9. Wei, C., He, Z., Zhang, Y. & Pei, W. (2002). Swarm directions embedded in fast evolutionary programming. *In Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 1278–1283, Honolulu, HI. Piscataway: IEEE.
10. Robinson, J., Sinton, S. & Rahmat-Samii, Y. (2002). Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. *In Proceedings IEEE International Symposium on Antennas and Propagation*, pp. 314–317, San Antonio, TX. Piscataway: IEEE.
11. Kaveh, A. & Malakoutirad, (2010). Hybride genetic algorithm and particle swarm optimization for the force method-based simultaneous analaysis and design. *Iranian Journal of Science & Technology, Transaction B: Engineering*, Vol. 34, No. B1, pp. 15-34
12. Kaveh, A. & Masoudi, S. (2013). Plate bending finite element analysis by the force method using ant colony optimization. *Iranian Journal of Science & Technology Transactions of Civil Engineering*, Vol. 37, No. C1, pp. 17-32
13. Clerc, M. (2006). *Particle swarm optimization*. London: ISTE.
14. Nagesh Kumar, D. & Janga Reddy, M. (2007). Multipurpose reservoir operation using particle swarm. *Optimization Journal of Water Resources Planning and Management*, Vol. 133, No. 3.
15. Shelokar, P. S., Patrick, S., Jayaraman, V. K. & Kulkarni, B. D. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, Vol. 188, pp. 129–142.
16. Dorigo, M. & Blum, C. (2005). Ant colony optimization theory: a survey. *Theoretical computer Science*, Vol. 344, No. 2–3, pp. 243–278.
17. Vanden Bergh, F. (2002). An analysis of particle swarm optimizers. PhD thesis, Depart- ment of Computer Science, University of Pretoria, South Africa.
18. Wang, H., Liu, Y., Li, C. & Zeng, S. (2007). A hybrid particle swarm algorithm with Cauchy mutation. *Proc. of the 2007 IEEE Swarm Intelligence Symposium*.
19. Liu, B., Wang, L., Jin, Y. H., Tang, F. & Huang, D. X. (2005). Improved particle swarm optimization combined with chaos. *Chaos Solitons and Fractals*, Vol. 25, pp. 1261–1271.