COMPARATIVE STUDY OF GA AND PSO ALGORITHM

Jaspreet Kaur[1], Satvir Singh[2], Sarabjeet Singh[3], Shivani Kakkar[4], Vijay Kumar Banga[5]
[1,2,3,4]SBS State Technical Campus, Ferozepur (Punjab), India, [5]Amritsar College of Engg. & Tech., Amritsar, (Punjab), India
Email:[1]er.jaspreetkaur1@yahoo.com, [2]Drsatvir.in@gmail.com, [3]sarabjeet_singh13@yahoo.com, [4]kakkarshivani@yahoo.in, [5]v_banga@rediffmail.com

Abstract— In the last few decades, both PSO and GA have gained much attention of the researchers as an effective methods for solving different optimization problems. PSO is one of the Swarm Intelligence techniques which is based on the collective behaviour of decentralized, self organized system. In this paper, we have presented a methodology of implementing PSO and GA in order to optimize a problem by iteratively improving a candidate solution. The algorithm was tested for well- known benchmark problems. The experiments were conducted on two different systems both for GA and PSO in order to choose the one which require less computing time. Being computationally intensive the execution speed of GA algorithm is very low. As a result, it was found that the PSO algorithm ran faster on both the systems. Moreover, System 2 possessing i5 processor accelerated the execution speed of the PSO algorithm and hence requires less computing time.

Index Terms—Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Swarm Intelligence (SI).

I. INTRODUCTION

The optimization problems can be easily solved by an innovative distributed paradigm known as Swarm Intelligence (SI). The concept of SI was introduced by Gerardo Beni and Jing Wang in 1989, who originally got inspired from the biological examples such as bird flocking, ant colonies, animal herding, fish schooling and bacterial growth. An attempt was made to design various algorithms or distributed problem solving devices based on the biological phenomena or systems.

PSO is a stochastic global optimization technique based on the social behaviour of bird flocking or fish schooling, developed by Eberhart and Kennedy in 1995 [1]. The fundamental idea is that each particle represents a potential solution which it updates according to two important kinds of information available in decision process. The first one (cognitive behaviour) is gained by its own experience, and the second one (social behaviour) is the experience gained from the neighbours, that is, they tried the choices itself and have the knowledge which choices their neighbours have outstand so far and how positive the best pattern of choices was. PSO has been used increasingly due to its several advantages like robustness, efficiency and simplicity. When compared with other stochastic algorithms it has been found that PSO requires less computational effort [2] [3]. Although PSO has shown its potential on many aspects for solving complex and difficult optimization problems, it still requires a long execution time to find solutions for large-scale engineering problems [4] [5].

One of the most important class of Evolutionary algorithms is Genetic algorithm (GA) inspired by evolutionary biology. The concept of GA was introduced by John Holland in the mid 1970 at University of Michigan [6].Genetic algorithm are categorized as global search heuristics that uses iterative process to obtain desired solutions. GA has been very efficient in many real world problems such as optimization, design and scheduling [7], power systems [8], data handling etc.

After this brief introduction, the rest of the paper is organized as follows: Section II gives a brief overview of PSO algorithm. In Section III implementation of PSO on CPU is presented. Section IV provides the brief overview of Genetic algorithm. Section V summarizes the performance evaluation of experimental results. The last section presents the conclusion of this paper and point out direction for future work.

II. PARTICLE SWARM OPTIMIZATION

PSO is a meta-heuristic algorithm works by having a swarm of particles. These particles are moved around in the search-space according to a few simple formulae. The movement of the particles are guided by their own best position in the search-space as well as the entire best known position [9] [10].The particles are initialized by a randomized velocity and position at the beginning of the search process, and then at each time step, the velocity and position of each particle is changed moving towards pbest and gbest location.

Acceleration coefficients are weighted by random terms to efficiently control the local search and convergence to the global optimum solution [11]. Separate random numbers are generated for acceleration towards pbest and gbest locations, respectively [12].

Consider the d-dimensional search space and ith particle in the swarm is represented by $X_i = (x_{i1}, x_{i2}, ...x_{id})$ and its velocity can be represented by another d-dimensional vector $V_i = (v_{i1}, v_{i2}, ... v_{id})$. Let the best position ever visited in the past by ith particle be denoted by $P_i = (p_{i1}, p_{i2}, ... p_{id})$. The personal best particle is denoted as $P_p = (p_{p1}, p_{p2}, ... P_{pd})$, and an overall best particle is denoted as $P_g = (p_{g1}, p_{g2}, ... p_{gd})$, where g and p are particle indices.

The velocity and position of the particle are updated by the given formula respectively:

$$V_{id}(t+1) = \chi\Big(V_{id}(t) + c_1\, r_1\big(P_{pbd}(t) - X_{id}(t)\big) + c_2\, r_2\big(P_{gbd}(t) - X_{id}(t)\big)\Big) \dots (1)$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \dots (2)$$

In equation (1), c_1 and c_2 are the learning factors which are non-negative constants. r_1 and r_2 are random numbers uniformly generated in the range [0,1]. Constriction factor is denoted by χ which is derived as:

$$\chi = \frac{2}{|\, 2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}$$

And, $\varphi = c1 + c2$

If ψ is set to 4.1, then $\chi = 0.729$ [13][14]. The particle velocity $V_{id} \in [-V_{min}, V_{max}]$, where V_{max} is a designated maximum velocity. If the velocity exceeds V_{max} in any coordinate it will be truncated to V_{max} to avoid search explosion. V_{max} is preset according to the objective optimization function. If it is too high, the particles could skip over good solutions and if too small, particles are explored too slowly and good solutions could not be found. $P_{pbd}(t)$ and $P_{gbd}(t)$ are the personal and global best position respectively.

III. IMPLEMENTATION OF PSO

PSO is a stochastic algorithm which requires lots of random numbers during the process of optimization. The performance of PSO algorithm is greatly affected by the quality of random numbers generated. The initial population is uniformly distributed over the entire search space. Fig 1 shows the flowchart of PSO.

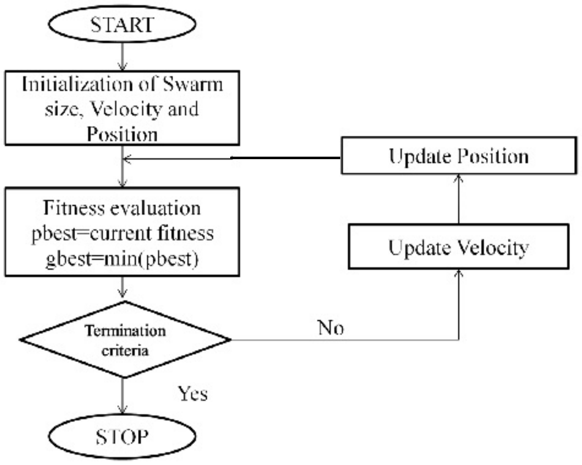

START
Initialization of Swarm size, Velocity and Position
Update Position
Fitness evaluation pbest=current fitness gbest=min(pbest)
Update Velocity
Termination criteria
No
Yes
STOP

Fig. 1: Flowchart of PSO

In this section, we describe the basic steps and code for implementing PSO algorithm in C language.

Step 1: Initialize three random number arrays of size N*D.

1) Past Position array
2) Present Position array
3) Velocity array

Step 2: Evaluate the fitness of each particle using test functions. Initially the current position and current fitness are the pbest position and fitness respectively.

a) The pbest position of the particle is calculated by comparing the past and present position matrix.
b) Since we are dealing with minimization problem, so a minimum value of pbest array is taken as gbest value.

Step 3: Update velocity and Position of the particle after every iteration using eq.(1)& eq.(2). For next iteration, the two new arrays generated after the update will become the present velocity and the present position array. The previous present position array will now become the past position array.

Step 4: Repeat the steps until the stopping criteria is met.

IV. GENETIC ALGORITHM

Genetic Algorithm is a heuristic search technique based on the evolutionary idea of genetics and natural selection. Like

PSO, GA also initiates its search process from the randomly generated population that evolve through consecutive generations. The flowchart of PSO is shown in Fig 2.

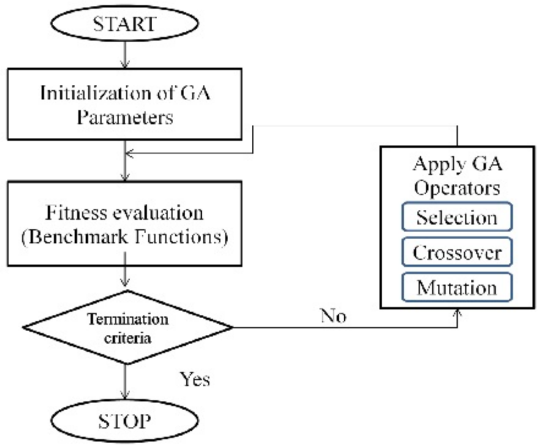


Fig. 2: Flowchart of GA

The GA steps are as follows: Initially the random population (feasible solution) of n chromosomes is generated and then the fitness value of each chromosome is calculated. Further the termination criterion is checked and GA operators such as selection, crossover, and mutation are applied to generate a new population.

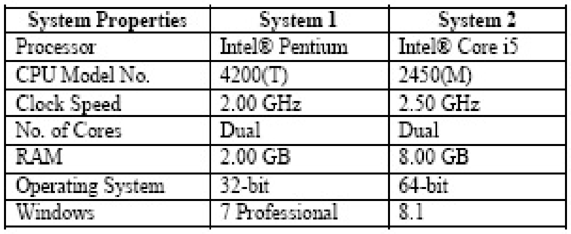
1. In the selection process, two parent chromosomes from a random population are selected by using Roulette wheel selection method.
2. Next step is to crossover the selected chromosomes to form new offspring. The exact same copy of selected chromosomes is produced if no crossover is performed.
3. With certain mutation probability, this offspring are mutated to a position and placed in the new generated population. The actual population is replaced by this new population.
4. Now, this newly generated population is used further in the iterative process.
5. The process is repeated until the termination criteria is met and best solution is not found.

V. PERFORMANCE EVALUATION

A. Experimental Setup

In order to make meaningful comparison, PSO Algorithm was implemented and tested on two different systems. The specifications of the system are listed in Table I. An average fitness computing time of the best solution is recorded after

30 trials for every function on each experimental setting. The experiment is evaluated using two different iteration size of 10,000 and 100,000.



System Properties	System 1	System 2
Processor	Intel® Pentium	Intel® Core i5
CPU Model No.	4200(T)	2450(M)
Clock Speed	2.00 GHz	2.50 GHz
No. of Cores	Dual	Dual
RAM	2.00 GB	8.00 GB
Operating System	32-bit	64-bit
Windows	7 Professional	8.1

Table I: System Specifications

B. Benchmark Functions

Six benchmark minimization functions were used in the
simulations. Out of which Schumer is Unimodal, while rest all are multimodal functions. Table II shows the list of benchmark functions that are used for optimization [15].

Benchmark Functions	Equation	Range	Global Minima	Type
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	-10< x_i <10	0	Multimodal
Schumer Stieglitz	$f_2(x) = \sum_{i=1}^{n} x_i^4$	-10< x_i <10	0	Unimodal
Quintic	$f_6(x) = \sum_{i=1}^{n} \lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4\rvert$	-10< x_i <10	0	Multimodal
Exponential	$f_4(x) = -exp\left(-0.5 \sum_{i=1}^{n} x_i^2\right)$	-1< x_i <1	1	Multimodal

Table II: Benchmark Functions

C. Result Analysis & Discussions

Table III and IV presents the computing time mean and standard deviation for both GA and PSO codes on each configuration. How fast the PSO algorithm than GA is measured in terms of speedup. Speedup 1 is the ratio of computing time of GA to the computing time of PSO for system 1 and Speed up 2 is the same ratio on system 2.

$$Speedup1 = \frac{GA1time(s)}{PSO1time(s)} \tag{3}$$

$$Speedup2 = \frac{GA2time(s)}{PSO2time(s)} \tag{4}$$

1. *Experiment 1:* The experiment is conducted by keeping population size of 32. The maximum number of iterations is first kept 10,000 and then 100,000 to gain valuable results. Results are recorded after a total of 30 runs for each benchmark function.

No. of Iterations	Functions	System 1				System 2				Speed up 1	Speed up 2
		GA1		PSO1		GA2		PSO2			
		Time(s)	Std Dev.	Time(s)	Std Dev.	Time(s)	Std Dev.	Time(s)	Std Dev.		
10,000	$f_1(x)$	6.688	0.012	4.567	0.109	5.234	0.150	3.940	0.237	1.464	1.328
	$f_2(x)$	6.928	0.034	5.151	0.174	5.856	0.464	4.322	0.182	1.345	1.355
	$f_3(x)$	9.513	0.033	6.314	0.049	7.805	0.509	5.577	0.212	1.507	1.399
	$f_4(x)$	6.354	0.033	3.869	0.051	5.294	0.150	3.054	0.091	1.642	1.733
100,000	$f_1(x)$	63.036	0.044	38.823	1.118	51.667	1.154	34.31	2.465	1.624	1.506
	$f_2(x)$	68.749	0.093	48.023	2.574	56.405	0.374	44.455	4.019	1.432	1.269
	$f_3(x)$	89.326	0.200	61.786	0.664	78.256	0.584	57.782	2.143	1.446	1.354
	$f_4(x)$	65.661	0.112	37.323	0.125	54.478	0.345	30.485	0.682	1.759	1.787

Table III: GA and PSO results with dimension size 32

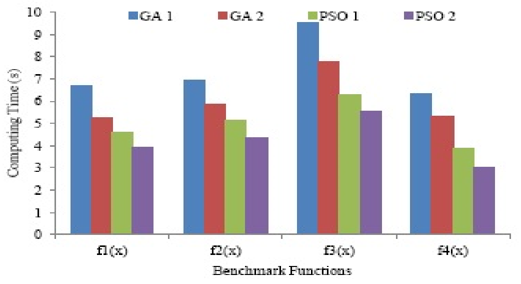


Fig. 3: Computing time with dimension size 32 and 10,000 iterations

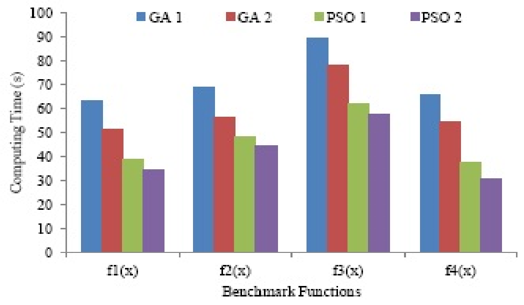


Fig. 4: Computing time with dimension size 32 and 100,000 iterations

2. Experiment 2: In this the population size is kept 64. Results are noted after 30 runs as in Experiment 1 keeping maximum number of iterations 10,000 and 100,000. Figure IV and V demonstrate that with the increase in dimension size, it requires more time to reach at optimal values.

No. of Iterations	Functions	System 1				System 2				Speed up 1	Speed up 2
		GA1		PSO1		GA2		PSO2			
		Time(s)	Std Dev.	Time(s)	Std Dev.	Time(s)	Std Dev.	Time(s)	Std Dev.		
10,000	$f_1(x)$	12.217	0.027	8.626	0.471	10.429	0.368	7.537	0.544	1.416	1.383
	$f_2(x)$	13.497	0.067	9.552	0.400	11.464	0.206	8.799	0.514	1.413	1.303
	$f_3(x)$	17.824	0.042	13.113	0.249	15.711	0.337	10.726	0.102	1.359	1.465
	$f_4(x)$	13.414	0.030	7.047	0.102	10.761	0.452	5.938	0.129	1.904	1.812
100,000	$f_1(x)$	119.791	0.065	75.837	0.743	102.849	0.804	64.512	3.484	1.579	1.594
	$f_2(x)$	131.497	0.172	87.169	2.264	113.136	0.769	76.435	5.055	1.508	1.480
	$f_3(x)$	178.286	0.229	123.807	0.722	153.438	1.484	112.539	4.311	1.440	1.363
	$f_4(x)$	125.544	0.128	74.579	0.229	106.289	2.392	61.687	2.956	1.683	1.723

Table IV: GA and PSO results with dimension size 64

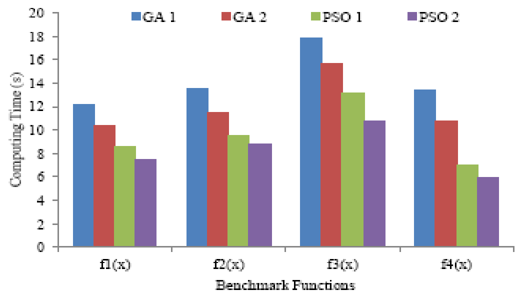


Fig. 5: Computing time with dimension size 64 and 10,000 iterations

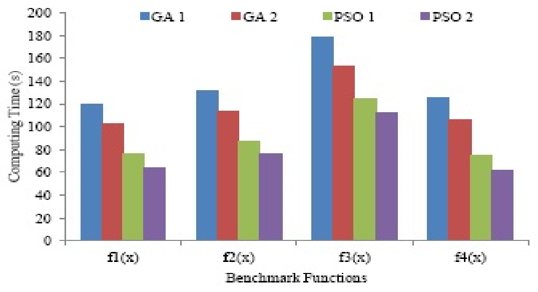


Fig. 6: Computing time with dimension size 64 and 10,000 iterations

In both the cases, After analyzing the data it has been found that exponential functions optimizes easily since it is computationally less intensive and quintic function require more time to find the solution. For all the functions PSO requires less execution time than GA to find a solution.

VI. CONCLUSION & FUTURE WORK

The execution speed of the designed system GA and PSO was compared on two different systems and it was concluded from the results that PSO reaches at target values in lesser iterations as compared to GA. Hence, quickly optimizes the problem. An average speed up of 1.3- 1.9 times than GA is obtained with PSO. On both the systems PSO outperforms than GA as it requires less computational effort. However,
System 2 shows better results for all benchmark functions with PSO algorithm since it possess i5 processor. In future,
Parallel implementation of PSO algorithm can be done to gain more efficient speedup. Further, PSO can be integrated with the Fuzzy Logic System for many applications involving wide range of classification.

REFERENCES

[1] R. Kennedy, J. ; Eberhart, "Particle swarm optimization," Neural Networks, 1995. Proceedings., *IEEE International Conference on Neural Networks, Perth, WA, Australia*,, vol. 4, pp. 1942– 1948, 1995.

[2] A. Engelbrecht, "Fundamentals of computational swarm intelligence, 2006," Hoboken: John Wiley & Sons, Ltd.

[3] N. Nedjah, L. dos Santos Coelho, and L. de Macedo Mourelle, Multiobjective swarm intelligent systems: theory & experiences. *Springer Science & Business Media, 2009*, vol. 261.

[4] A. Kumar, A. Khosla, J. S. Saini, and S. Singh, "Meta-heuristic range based node localization algorithm for wireless sensor networks," in *Localization and GNSS (ICL-GNSS), 2012 International Conference on. IEEE*, 2012, pp. 1–7.

[5] S. Singh, S. Shivangna, and E. Mittal, "Range based wireless sensor node localization using pso and bbo and its variants," in Communication Systems and Network Technologies (CSNT), *2013 International Conference on. IEEE*, 2013, pp. 309–315.

[6] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[7] O. Maimon and D. Braha, "A genetic algorithm approach to scheduling pcbs on a single machine," *International Journal of Production Research*, vol. 36, no. 3, pp. 761–784, 1998.

[8] J. Zhang, H. Chung, and W.-L. Lo, "Pseudocoevolutionary genetic algorithms for power electronic circuits optimization," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 36, no. 4, pp. 590–598, 2006.

[9] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," 2002.

[10]J. Kennedy, J. F. Kennedy, and R. C. Eberhart, *Swarm intelligence*. Morgan Kaufmann, 2001.

[11] A. Ratnaweera, S. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *Evolutionary*

Computation, IEEE Transactions on, vol. 8, no. 3, pp. 240–255, 2004.

[12]S. Singh, J. Kaur, and R. S. Sinha, "A comprehensive survey on various evolutionary algorithms on gpu," 2014.

[13] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 1, pp. 58–73, 2002.

[14]R. Mendes, "Population topologies and their influence in particle swarm performance," Ph.D. dissertation, Citeseer, 2004.

[15] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems, int," *Journal of Mathematical Modelling and Numerical Optimisation,* vol. 4, no. 2, pp. 150–194.