Particle Swarm Optimization and Its Hybrids

M. Fikret Ercan and Xiang Li

***Abstract*—Particle Swarm Optimization (PSO) algorithm has been widely used in various engineering problems because of its simplicity and efficiency. However, the PSO has a problem of premature convergence, due to the lack of diversity. The performance of the PSO algorithm can be further improved by hybrid techniques. There are numerous hybrid PSO algorithms published in the literature where researchers combine the benefits of PSO with other heuristic algorithms such as genetic algorithm (GA), ant colony optimization (ACO) just to name a few. In this paper, we present some of the commonly used hybrid PSO algorithms and study the performance of them through typical nonlinear optimization problems.**

***Index Terms*—Particle swarm optimization, hybrid PSO, swarm intelligence**

I. Introduction

Particle swarm optimisation (PSO) received significant attention during the last decade [1],[2]. There are many other metaheuristics known to research community though efficiency and simplicity of PSO make it a favourable choice in many applications such as power systems, transportation, scheduling, computing and so on [3], [4], [5]. Particle Swarm Optimization (PSO) is a swarm intelligence technique proposed by Kennedy and Eberhart [6]. It mimics the behaviour of flying birds and their communication mechanism to solve optimization problems and it is based on the constructive cooperation with a group of virtual particles. The algorithm is not computationally intensive and it has a few parameters to tune.

PSO algorithm is robust and it has a well global exploration capability though, it also has the tendency of being trapped in local minima and slow convergence. The performance of PSO can be improved by using hybrid techniques as it is shown with vast amount of research in literature (see for instance [7], [8]). Hybrid algorithms typically combine the well-known heuristics with PSO. Genetic algorithm (GA) and PSO hybrid is one popular approach. For example, D.H. Kim et. al. show PSO-GA hybrid algorithm for the optimal tuning of proportional integral derivative (PID) controllers which is widely used in industrial systems [9]. In another study, author demonstrates application of PSO- tabu search (TS) and PSO simulated annealing (SA) hybrid algorithms into multiprocessor task scheduling problem where PS-SA hybrid considerably outperforms other methods [10]. PSO and ant colony optimization (ACO) hybrid has also been implemented in data mining and classification problem [11].

In most of the research published earlier, the hybridisation of PSO with other well-known metaheuristics show significant performance improvement when compared to PSO alone though it is usually achieved at the expense of increased computational complexity On the other hand, hybridisation with rather simpler heuristics demonstrated significant performance improvement with lesser computational complexity. In our earlier study, we reported a hybrid PSO local line search algorithm [12]. The local line search usually requires less iteration and it has the advantage of strong local search and fast convergence. However, its limitation in global search is a critical drawback. The hybrid PSO-Line search algorithm exploits the advantages of line search and PSO algorithms by combining global search ability of standard PSO and local search ability of the line search algorithm.

In this paper, we present an empirical study of the popular hybrid PSO algorithms and their performance. In the following, we give a brief description and the operation principles of PSO and the hybrid PSO algorithms chosen for the study. It is followed with the results of our computational study. There are many hybrid PSO algorithms published in literature. Here, we narrow our scope to a selected few by considering the application variety and their popularity.

PSO algorithm

Standard PSO is a stochastic search algorithm in multimodal search space, emerging from simulations of dynamic systems such as bird flocks and fish swarms. The idea of PSO is to have a swarm of particles flying through a multidimensional search space, looking for the global optimum. By exchanging information, particles can influence each others' movements. Each particle retains an individual (or "cognitive") memory of the best position it has visited, as well as a global (or "social") memory of the best position visited by all the particles in the swarm. A particle calculates its next position based on a combination of its last movement vector, the individual and global memories, and a random component.

Each particle is initialized at a random position in search space. The position of particle i is given by a vector, $x_i = (x_{i1}, x_{i2},..., x_{iD})$ where D indicates the dimensionality of the problem. Its velocity is given by the vector $v_i = (v_{i1}, v_{i2},..., v_{iD})$. Two kinds of memory are utilised and they influence the movement of the particles. In the cognitive memory, $p_i = (p_{i1}, p_{i2},..., p_{iD})$, the best previous position visited by each individual particle is stored. On the other hand, the position of the best point in search space visited by all the particles so far is stored in vector, $p_g = (p_{g1}, p_{g2},..., p_{gD})$, which is also called "social memory". The particle velocities are updated

Manuscript received July 24, 2012; revised September 15, 2012.

M. F. Ercan is with the School of Electrical and Electronic Engineering, Singapore Polytechnic, Singapore 139651 (e-mail: mfercan@sp.edu.sg).

X. Li is with the School of Information Science and Engineering, Central South University, Changsha, 410083, China (e-mail: xl_huse@126.com).

according to the equation given as:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i - x_i(t)) + c_2 \cdot r_2 \cdot (p_g - x_i(t)) \quad (1)$$

In equation (1), w is the initial weight, which is a weighting factor for previous velocity. c_1 and c_2 are positive constants called "cognitive" and "social" parameter weightings that influence two different swarm memories. r_1 and r_2 are random numbers between 0 and 1. A restriction constant $V_{\max}$ is used to control the velocity of particles. Velocities exceeding the threshold set by $V_{\max}$ are set back to this value. The initial weight w can be implemented either as a constant or as a variable which changes linearly over time. Typically, a start and end value is set and for each iteration a new value of w is calculated as shown in equation (2).

$$w = w_{start} - \frac{w_{start} - w_{end}}{I_{Max}} . I_n \quad (2)$$

Here, w_{start} is the initial and w_{end} is the terminal value of I_n. In addition, I_{Max} stands for the current iteration and is the maximum number of iterations for the PSO. Typically a large starting value is used, causing the swarm to perform more global search with large movements at the beginning and shifting towards smaller movements and fine tuning towards the end of the optimization process. After the velocity vector is calculated the positions of the particles are updated according to the following equation:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (3)$$

Typically, a maximum number of iteration is defined as termination condition of the PSO algorithm. Alternatively, a combination of other conditions may also be introduced depending on the specific requirements of the application.

II. Hybrid PSO algorithms

PSO-GA and PSO SA: The basic idea of the hybrid algorithms presented here have two major operations first running PSO algorithm and obtaining a global best solution and then improving the result with GA [9]. Typical application of PSO-SA hybrid involves an initial search with PSO and refining result with simulated annealing. SA introduces a probability to avoid becoming trapped in a local minimum. The pseudo codes shown in Fig. 1 describe the PSO-GA and PSO-SA hybrid algorithms:

```
Initialize swarm with random positions and velocities;
    begin
    initialize PSO and SA;
    while (termination !=true)
    do{generate swarm;
    compute and find best Pg;}
    set particle that gives best Pg as initial solution to GA;
    while (!=Stopping criterion)
    do{Crossover the GBest;
    Mutation the GBest
    Evaluate the fitness of each chromosome;}
    end.
```

(a)

```
Initialize swarm with random positions and velocities;
    begin
    initialize PSO and SA;
    while (termination !=true)
    do{generate swarm;
    compute and find best Pg;}
    set particle that gives best Pg as initial solution to SA;
    while (Tcurrent>Temp_end)
    do{
    generate solution;
    evaluate and update best solution and temperature; }
    end.
```

(b)

Fig. 1. (a) PSO-GA (b) PSO-SA hybrid algorithms

PSO Line search: In our proposed PSO line search hybrid algorithm, some particles from the current generation are selected to form a sub swarm (subswarm-1) and joined in Armijo line search. These particles may achieve a sufficient increase in their fitness. In that case, we let the swarm parameter p_g immediately reflect the improvement of fitness achieved by these particles. The rest of the swarm (subswarm-2) execute the PSO algorithm. They are also allowed to update p_g. Finally, these two sub swarms are merged into a single swarm and employed for the next iteration. This procedure is repeated until a termination criterion is reached. A more detailed and comprehensive report about this algorithm can be found in [12]. The pseudo code in Fig. 2 shows the operations of PSO-Line search hybrid:

```
Initialize swarm size and positions and Armijo parameters: λ0 , β , α ;
while (number of generations< I_Max ) {
Select P_N particles from swarm as subswarm-1; (we employ random selection)
for each particle i in subswarm-1 {
if (|| Gradient ( x_i ) || < G_0 ) {
Execute Armijo line search using x_i as an initial point, then gain new x_i , and maintain v_i unchanged; }
endif
} endfor
evaluate subswarm-1 and update p_i and p_g ;
for subswarm-2, execute S-PSO according to equation (1);
evaluate subswarm-2 and update p_i and p_g ;
merge subswarm-1 and subswarm-2 into a whole swarm;}
endwhile
```

Fig. 2. PSO-Line search hybrid algorithm

III. Performance

A set of nonlinear functions are used to evaluate algorithm performance. These functions are Sphere (f_{Sh}), Rosenbrock (f_{Ro}), Rastrigrin (f_{Ra}) and Griewank (f_{Gr}). Details of them are given in [12]. Parameter setting for PSO was as follows. The inertial weight w was decreased from $w_{start} = 0.9$ to $w_{end} = 0.4$ linearly for $I_n = 1,...,0.7 \times I_{Max}$ and for the remaining iterations, w remains at 0.4. The acceleration constants are set as $c_1 = c_2 = 2$. In the case of GA, the crossover rate and the mutation rate were chosen as 0.6 and 0.2 following the commonly used norms. For the simulated annealing, a simple cooling strategy is employed. Temperature is decreased through an exponential manner with $T_i = \lambda T_{i-1}$ where $\lambda < 1$. In our implementation value for λ is selected as 0.998 after repetitive experiments. The initial temperature is selected from the following formula:

$$T_o = \frac{\overline{\Delta E}_{avg}}{\ln(x_0)}$$

Here $\overline{\Delta E}_{avg}$ is the average increase in the cost for a number of random transitions. Initial acceptance ratio, x_0, is defined as the number of accepted transitions divided by the number of proposed transitions. These parameters estimated after 50 randomly permuted solution of the initial solution. Parameter settings for Armijo line search tune two key components. In our algorithm, if it is anticipated that the fitness of a particle, which is selected to join the line search, is not going to be improved considerably in the future generations, we try to exclude that particle in line search. In other words, if a selected particle locates at a steep slope then it should start the line search. Parameter G_0, shown in the pseudo code in Figure 2, is used for this purpose. Another parameter λ_0 is the initial step size required for Armijo line search. In the following experiments, both G_0 and λ_0 are tuned so that approximately the same evaluation time is given to the test functions. Other Armijo parameters are fixed as $\alpha = 0.001$, $\beta = 0.1$ and number of particle selected for line search is 4. We have employed a fixed number of iterations as stopping criterion. All the algorithms perform 80 iterations.

TABLE I: Functions with DIMENTION 10 Using Algorithms with Particle Size 10 and Generation Number 80.

Function/Optimal result	BEST	MEAN	WORST	STD	Algorithm
$f_{Sh}/0$	10.99746	79.832484	250.21541	57.91012	PSO
	0.145121	*2.928731*	*7.9081855*	*1.384219*	PSO-GA
	0.592131	*4.562813*	*11.63713*	*1.163739*	PSO-SA
	0.151992	*3.274787*	*8.813421*	*1.921044*	PSO_LS G_0=20 λ_0= 0.1
$f_{Ro}/0$	6.208669	24.618283	81.752671	22.37126	PSO
	0.891378	*4.0519841*	*12.508832*	*1.953121*	PSO-GA
	1.342015	*7.645892*	*10.142873*	*1.735610*	PSO-SA
	0.930739	*6.109603*	*9.752152*	*1.87612*	PSO_LS G_0=20 λ_0= 0.001
$f_{Ra}/0$	17.9758	41.686909	69.701227	11.61018	PSO
	6.12873	*28.001763*	*60.056141*	*10.17821*	PSO-GA
	6.01736	*30.452548*	*62.873732*	11.73621	PSO-SA
	7.34792	*29.921405*	*52.664531*	*10.10675*	PSO_LS G_0=20 λ_0= 0.01
$f_{Gr}/0$	1.042606	1.829192	4.409106	0.68367	PSO
	0.628651	1.886202	*3.363891*	*0.874526*	PSO-GA
	0.709633	2. 008102	4.126439	1.72666	PSO-SA
	1.374584	3.803974	7.918195	1.448149	PSO_LS G_0=0 λ_0=1

From the results presented in Table I, it is obvious that overall PSO-GA algorithm outperforms other hybrid algorithms that we have experimented. For the PSO-line search algorithm it is expected that when number of particles is increased, the number of function evaluations will also increase significantly. However, at present, we have not concluded the best particle value for different complexity of problems. From the results above, we observe that PSO-LS is unable to demonstrate a notably improved performance for the function f_{Gr}. Even for the increased number of particles in line search (from 2 to 6) we do not observe a major improvement. However, for the same function PSO-GA gives almost double the better performance. Overall, PSO-GA and PSO-SA performances rather close. Only in f_{Ra} PSO-SA gave better result than PSO-GA.

IV. Summary

This paper presented our initial findings on the performance comparison of various hybrid PSO algorithms that are commonly used in practice. This study was conducted using a set of non-linear functions compare its performance with standard PSO. Among the hybrids algorithm PSO-GA out performed others whereas PSO line search was the least time consuming. In our future study, we intend to conduct more experiments with a larger spectrum of hybrid PSO algorithms and more detailed performance study with various parameters of individual algorithms.

References

[1] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part I: Background and development," *Natural Computing*, vol. 6, no. 4, pp. 46-484, 2007.

[2] A. Banks, J. Vincent, and C. Anyakoha, "Review of particle swarm optimization. Part II: Hybridization, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Computing*, vol. 7, no. 1, pp. 109-124, 2008.
[3] M. R. Alrashidia and M. E. El-Hawarya, "A survey of particle swarm optimization applications in power system operations," *Electric Power Components and Systems*, vol. 34, no. 12, pp. 1349-1357, 2006.
[4] Y. Li and X. Chen, "Mobile robot navigation using particle swarm optimization and adaptive NN," *Lecture Notes in Computer Science*, vol. 3612, pp. 628-631, 2005.
[5] G. Xu, H. Sun, and W. Yang, "Particle swarm optimization for road extraction in SAR images," *Lecture Notes in Control and Information Sciences*, vol. 345, pp. 392 – 401, 2006.
[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proc. of IEEE Int. Conf. on Neural Networks*, 1995, pp. 1942-1948.
[7] G. Yang, D. Chen, and G. Zhou, "A new hybrid algorithm of particle swarm optimization," *Lecture Notes in Bioinformatics*, vol. 4115, pp. 50-60, 2006.
[8] Q. Zhang, C. Li, Y. Liu, and L. Kang, "Fast multi-swarm optimization with cauchy mutation and crossover operation," *Lecture Notes in Computer Science*, vol. 4683, pp. 344-352, 2007.
[9] D. H. Kim, A. Abraham, and K. Hirota, "Hybrid genetic: Particle swarm optimization algorithm," *Studies in Computational Intelligence*, vol. 75, pp. 147–170, 2007.
[10] M. F. Ercan, "Hybrid particle swarm optimization and other metaheuristic methods in flow shop scheduling problem," in *Aleksandar Lazinica Particle Swarm Optimization*. 2009, pp. 155-168.
[11] N. Holden, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," *Proceedings of Swarm Intelligence Symposium*. 2005, pp. 100-107.
[12] X. Liang, X. Li, and M. F. Ercan, "A PSO − Line search hybrid algorithm," *Lecture Notes in Computer Science*, vol. 5593, pp. 556–565, 2009.