A Modified PSO Algorithm With Line Search

Guanghui Zeng
Department of Electronic and Information Engineering
Guangzhou City Construction College
Conghua, China

Yuewen Jiang
School of Physics and Optical Information Sciences
Jiaying University
Meizhou, China

Abstract—Recently Particle Swarm Optimization (PSO) algorithm gained popularity and employed in many engineering applications because of its simplicity and efficiency. The performance of the PSO algorithm can further be improved by using hybrid techniques. There are various hybrid PSO algorithms published in the literature where researchers combine the benefits of PSO with other heuristics algorithms. In this paper, we propose a cooperative line search particle swarm optimization (CLS-PSO) algorithm by integrating local line search technique and basic PSO (B-PSO). The performance of the proposed hybrid algorithm, examined through four typically nonlinear optimization problems, is reported. Our experimental results show that CLS-PSO outperforms basic PSO.

Keywords-PSO; line search; hybrid algorithm; testing problem

I. INTRODUCTION

Many engineering applications such as automation, chemical engineering, image processing, communication, mechanical design etc., involve in finding an optimal solution [1, 2]. There are many algorithms introduced in the literature to find optimum solutions to problems with high computational complexity. However, developing highly efficient optimization algorithms is still attracting the attention of research community as such algorithms have many applications in various engineering fields.

Particle swarm optimization (PSO) is another evolutionary computation technique proposed by Kennedy and Eberhart [3]. It mimics the behavior of flying birds and their communication mechanism to solve optimization problems [3]. It is based on a constructive cooperation between particles in contrast to survival of the fittest approach used in other evolutionary methods. PSO has many advantages therefore algorithm has gained popularity recently [1, 2, 5, 6] and found applications in many practical engineering problems [1, 2]. The algorithm is simple, fast and very easy to code. It is not computationally intensive in terms of memory requirements and time. Furthermore, it has a few parameters to tune.

Although, PSO is very robust and has a well global exploration capability, it has the tendency of being trapped in local minima and slow convergence. The performance of PSO can be improved by using hybrid techniques (see for instance [5, 6]). These algorithms basically combine well known heuristics with PSO. Their results show significant performance improvement though this is at the expense of increased computational complexity and/or complexity of the basic algorithm. On the other hand, it is well known that local line search usually requires less iteration and it has the advantage of strong local search and fast convergence. More explicitly a latter iteration in local line search algorithm can warranty a better result than the former iteration though its inability of global exploration is a critical drawback. In this research, we integrate local line search and basic PSO algorithms and propose a new hybrid PSO algorithm. The new hybrid algorithm, cooperative line search PSO (CLS-PSO), benefits from global search ability of the PSO and the local search ability of the line search algorithm. Hence, we expect that the hybrid algorithm will improve the performance of the basic PSO algorithm. The following sections present the basic idea of CLS-PSO algorithm and analysis of the results.

II. BASIC PSO ALGORITHM

PSO is initialized with a population of random solutions (particles). Each particle has two states, the current position p and the current velocity v. Particle has an ability of memory to the best position (Pi) itself experienced and the best position (Pg) swarm experienced. At each generation, the velocity and position of each particle is updated using following formulas,

$$v_i = w * v_{i+1} + c_1 * r_1 * (P_i - p_i) + c_2 * r_2 * (Pg - pi)$$

$$p_{i+1} = p_i + v_{i+1}$$

Where w is called the inertial weight, c1 and c2 are the acceleration constants, r1 and r2 are the random numbers uniformly generated from [0, 1]. A limit velocity called vmax is imposed on particles. If calculated velocity of a particle exceeds this value, it will be reset to the maximum velocity.

III. LOCAL LINE SEARCH PSO (CLS-PSO)

As mentioned earlier, PSO have strong global exploration ability while having low convergence. In CLS-PSO algorithm, we select certain number of particles from the current generation and let them join an Armijo line search. These particles may achieve a sufficient increase in their fitness. In that case, we let the swarm parameter Pg immediately reflect the change of fitness achieved by these particles. Rest of the swarm executes basic PSO algorithm, they are also allowed to update Pg. Finally, two sub-swarms are merged into a single swarm and get ready for the next iteration. This procedure is repeated and cycled. The CLS-PSO algorithm is summarized with the following pseudo code:

initialization:

{ swarm scale: SIZE;

978-1-4244-5392-4/10/$26.00 ©2010 IEEE

initial states of particles: Vi, Xi;

running parameters: c1, c2, w;

number of particles expected for line search: PN;

constant parameter: NGrad;

Armijo parameters: StepLen, β, α;

maximal number of generations: GenNum

}

while (number of generations < GenNum) {

① from swarm, select PN particles according to certain strategy as sub-swarm-1; (here, we employ random selection)

② for each particle i in sub-swarm-1 {

if (|| FunGradient(Xi) || < NGrad) {

execute Armijo line search using Xi as an initial point, then gain new Xi, and maintain Vi unchanged; }

}

③ update Pg;

④ sub-swarm-2 consisting of rest of swarm, executes basic PSO according to equation (1);

⑤ evaluate sub-swarm-2;

⑥ update Pi of each particle in sub-swarm-2 and Pg;

⑦ merge sub-swarm-1 and sub-swarm-2 into a whole swarm;

}

IV. NUMERICAL EXPERIMENTS AND RESULTS

Four typical nonlinear functions are used to evaluate algorithm performance. These functions are defined as follows:

Sphere function:

$$f_{Sh}(\vec{x}) = \sum_{i=1}^{n} x_i^2$$

unimodal, global minimum: f(x)=0,x_i=0.

Rosenbrock function:

$$f_{Ro}(\vec{x}) = \sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$

unimodal, global minimum: f(x)=0,x_i=0.

Rastrigrin function:

$$f_{Ra}(\vec{x}) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$$

multimodal, global minimum: f(x)=0,x_i=0.

Griewank function:

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

multimodal, global minimum: f(x)=0,x_i=0.

Parameter setting for PSO: We decrease the inertial weight w from 0.9 to 0.4 linearly, for the number of generations from 1 to [0.7*GenNum]. GenNum is the maximum number of generations specified by user for the algorithm. For the remaining generations w remains as 0.4. We set the acceleration constants as c1=c2=2.0.

Parameter setting for Armijo line search: In our algorithm, if it is anticipated that the fitness of a particle, selected to join the line search, is not going to be improved significantly in next generations, we try to avoid that particle to take part in the line search. In other words, if a selected particle locates at a steep slope it should start the line search. A constant parameter, NGrad, shown in the above pseudo code is used to achieve this goal. Parameter NGrad can be tuned by user for a particular experiment. The parameter StepLen is another constant and it is the initial step length required for Armijo line search. This parameter can be tuned as well. In our experiments, both NGrad and StepLen are used to achieve about the same evaluation times for test functions during execution of CLS-PSO. Other parameters in Armijo equation are α =0.001, β =0.1 and their values remain unchanged in our experiments.

For each test function, the initialization intervals and vmax values are shown in Table 1.

TABLE I. INITIALIZATION INTERVALS AND MAXIMUM VELOCITY.

function	Initialization interval	vmax,d
fSp	[-100, 100]	100
fRo	[-2.048. 2.048]	2.048
fRa	[-5.12, 5.12]	5.12
fGr	[-600, 600]	600

We conduct three different set of experiments. For each set, we assign different values for SIZE, GenNum and dimension of functions. For each function, the statistical results are obtained by running the algorithm of B-PSO or CLS-PSO fifty times. Parameters NGrad and StepLen determine the average evaluation times (FNum) of functions. We adjust these parameters so that for the same function in a set of experiments evaluation times are approximately similar. In addition, in order to observe, how different number of particles that join the line search affects the performance of CLS-PSO, we try to change the values of PN. Table 2, 3 and 4 show results obtained for three sets of experiments with different parameters settings. In the tables, normal bold numbers show the results of B-PSO. The results of CLS-PSO, which are better than that of B-PSO, are marked with italic and the best results among them are highlighted with bold. For two dimensions case, CLS-PSO delivers the best results for functions fSh, fRa and fGr when the value of PN is 1, and for fRo, when PN is 3 (see Table 2). For ten dimensions case, when PN is 6 we obtained the best results for fSh, fRa and fRo except for fGr (see Table 3). For thirty dimensions, the best results are obtained when the value of PN is 8, except for fGr (see Table 4). Approximately, equal computation times are allocated for all functions in same set of experiments. It is clearly observed that the performance of CLS-PSO outperforms that of B-PSO except for fGr . However at present, we have not concluded the best PN value for different complexity of problems.

The reason why CLS-PSO is unable to demonstrate a notably improved performance for the function fGr is not clear. Even for the increased PN value, we do not observe a major improvement. One possible reason could be the experiment settings. We need to study further and conduct more experiments for the answer. However, from the results presented it is obvious that over all CLS-PSO outperforms the basic PSO algorithm.

V. CONCLUSIONS

In this paper, we proposed cooperative line search PSO hybrid evolutionary algorithm (CLS-PSO), and compare its performance with that of B-PSO by numerical experiments. From the statistical results, we conclude that proposed algorithm outperforms basic PSO. In our future study, we will conduct more experiments to understand the behavior of CLS-PSO. In addition, the collaboration methods between line search and PSO, effective selection strategies for particles joining the line search and its implications on the result need to be studied further.

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. 467-484, 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] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.

[4] Y. H. Shi and R. C. Eberhart, “Empirical Study of Particle Swarm Optimization,” Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 1945-1950, 1999.

[5] G. Yang, D. Chen, and G. Zhou, “A New Hybrid Algorithm of Particle Swarm Optimization”, LNBI, vol. 4115, pp. 50-60, 2006.

[6] Q. Zhang, C. Li, Y. Liu, et al, “Fast Multi-swarm Optimization with Cauchy Mutation and Crossover Operation,” LNCS vol. 4683, pp. 344-352, 2007.

Table 2. Functions with 2 DIM using algorithms with SIZE=5 and GenNum=50

FUN/OPT	BEST	MEDIAN	MEAN	WORST	STD	FNum	PN	VERSION	
$f_{Sh}/0$	0.000203	0.009949	0.050915	0.834574	0.125179	255		B-PSO	
	0.000024	0.005982	0.027989	0.413848	0.065827	249	1	CLS-PSO	NGrad=2 StepLen=1
	0.000031	0.008033	0.02659	0.440727	0.066135	242	2		
	0.00059	0.017284	0.048898	0.394076	0.082348	226	3		
$f_{Ro}/0$	0.000104	0.037134	0.178293	4.22069	0.6163	255		B-PSO	
	0	0.002306	0.012047	0.161764	0.028923	254	1	CLS-PSO	NGrad=2 StepLen=0.01
	0.000001	0.00016	0.006756	0.24979	0.03612	251	2		
	0.000001	0.000122	0.000897	0.022802	0.003311	239	3		
$f_{Ra}/0$	0.000069	0.998391	0.880693	4.97689	0.870117	255		B-PSO	
	0.000135	0.995223	0.86459	2.952167	0.771634	255	1	CLS-PSO	NGrad=2 StepLen=0.01
	0.002443	1.004706	0.905111	2.204677	0.636771	254	2		
	0.000093	0.997247	1.013054	3.989579	0.954166	251	3		
$f_{Gr}/0$	0.020092	0.133698	0.149837	0.365341	0.089461	255		B-PSO	
	0.000003	0.073973	0.117988	0.417868	0.108264	255	1	CLS-PSO	NGrad=0 StepLen=1
	0.007687	0.122168	0.17001	0.801768	0.155928	255	2		
	0.001589	0.119108	0.161414	0.713181	0.149471	255	3		

Table 3. Functions with 10 DIM using algorithms with SIZE=10 and GenNum=80

FUN/OPT	BEST	MEDIAN	MEAN	WORST	STD	FNum	PN	VERSION	
$f_{Sh}/0$	10.99746	68.294035	79.832484	250.21541	57.91012	810		B-PSO	
	2.604126	14.448428	17.206827	83.301085	14.61279	808	1	CLS-PSO	NGrad=20 StepLen=0.1
	1.138475	7.670247	8.691129	25.87961	5.105647	803	2		
	0.151992	2.875198	3.274787	8.8134	1.921044	753	4		
	0.591864	**2.649094**	**2.880512**	**8.464009**	**1.523916**	669	6		
$f_{Ro}/0$	6.208669	13.465615	24.618283	81.752671	22.37126	810		B-PSO	
	4.332262	8.786142	9.057704	17.426607	2.08105	814	1	CLS-PSO	NGrad=20 StepLen=0.001
	2.289102	8.101985	7.938878	10.37776	**1.537704**	817	2		
	0.930739	6.309097	6.109603	9.752152	1.87612	814	4		
	0.047277	**5.333849**	**4.959048**	**8.830937**	2.287102	805	6		
$f_{Ra}/0$	17.9758	43.49918	41.686909	69.701227	11.61018	810		B-PSO	
	11.05158	32.668272	33.919513	64.147091	11.47117	810	1	CLS-PSO	NGrad=20 StepLen=0.01
	9.714402	29.986856	31.710038	61.714373	12.6663	919	2		
	7.34792	29.558088	29.921405	52.664531	10.10675	1044	4		
	10.00827	**23.424202**	**24.875928**	**52.253293**	**8.993493**	1190	6		
$f_{Gr}/0$	1.042606	1.717609	1.829192	4.409106	0.68367	810		B-PSO	
	1.099857	**1.678122**	1.886202	**3.719182**	**0.674526**	810	1	CLS-PSO	NGrad=0 StepLen=1
	1.151146	2.110503	2.440084	7.237958	1.175986	810	2		
	1.374584	3.755625	3.803974	7.918195	1.448149	810	4		
	2.216543	6.233637	6.988493	16.516065	3.206142	810	6		

Table 4. Functions with 30 DIM using algorithms with SIZE=15 and GenNum=100.

FUN/OPT	BEST	MEDIAN	MEAN	WORST	STD	FNum	PN	VERSION	
$f_{Sh}/0$	1408.113	3773.244368	3851.199715	6538.82719	1180.174	1515		B-PSO	
	78.72999	334.841966	431.591709	1471.71128	314.3907	1514	1	CLS-PSO	NGrad=30 StepLen=0.1
	58.47356	128.987634	163.417036	793.621874	120.1027	1510	2		
	19.75363	63.036612	65.69403	145.092442	22.52022	1468	4		
	19.29957	**49.449917**	51.159679	86.857909	16.53508	1389	6		
	20.70181	49.614105	**49.975313**	**80.919829**	**13.68148**	1261	8		
$f_{Ro}/0$	184.1421	349.188669	361.497842	736.604201	101.1722	1515		B-PSO	
	31.93366	62.464877	66.933892	208.118929	31.20958	1531	1	CLS-PSO	NGrad=30 StepLen=0.001
	28.14126	40.454856	48.300594	109.916027	19.49501	1537	2		
	22.39567	29.739173	30.35641	78.446942	7.140461	1539	4		
	24.23159	28.57671	28.580692	30.604132	1.425963	1521	6		
	23.76544	**27.11673**	**27.104276**	**30.328539**	**1.369086**	1478	8		
$f_{Ra}/0$	157.2403	228.941886	230.901811	301.384898	34.33205	1515		B-PSO	
	101.9565	196.916347	199.15285	290.761195	41.21422	1515	1	CLS-PSO	NGrad=30 StepLen=0.001
	101.7126	174.00689	179.475336	280.838706	43.51307	1515	2		
	77.81324	180.136093	175.316802	242.476217	37.03607	1514	4		
	77.1869	163.662048	163.890833	233.429565	**33.77958**	1514	6		
	70.83228	**150.574619**	**150.955867**	**224.989656**	34.1704	1514	8		
$f_{Gr}/0$	17.93947	36.763406	40.103411	85.620382	14.89935	1515		B-PSO	
	16.26624	41.678887	41.74746	**63.674032**	**10.96972**	1515	1	CLS-PSO	NGrad=0 StepLen=1
	21.59679	46.588436	44.20316	70.033391	13.23021	1515	2		
	18.60879	55.441435	53.597909	100.432082	18.2983	1515	4		
	27.5395	66.335886	68.060567	113.452199	18.01507	1515	6		
	44.8756	80.600421	84.663288	136.311632	21.99389	1515	8		