[Cooperative Group Optimization] http://www.wiomax.com/optimization

Adaptive Particle Swarm Optimization on Individual Level

Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang

Institute of Microelectronics, Tsinghua University, Beijing 100084, P.R.China

Email: xiexiaofeng@tsinghua.org.cn

Abstract - An adaptive particle swarm optimization (PSO) on individual level is presented. By analyzing the social model of PSO, a replacing criterion based on the diversity of fitness between current particle and the best historical experience is introduced to maintain the social attribution of swarm adaptively by taking off inactive particles. The testing of three benchmark functions indicates it improves the average performance effectively.

Key words: adaptive particle swarm optimization, evolutionary computation, social model

1. Introduction

Modern heuristic algorithms are considered as practical tools for nonlinear optimization problems, which do not require that the objective function to be differentiable or be continuous. The particle swarm optimization (PSO) algorithm [1] is an evolutionary computation technique, which is inspired by social behavior of swarms. It has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement [2].

Work presented in [3] describes the complex task of parameter selection in the PSO model. PSO has been proved to be a competitor to the standard genetic algorithm (GA). Comparisons between PSO and GA were done with regards to performance by Angeline [4], which points out that the PSO performs well in the early iterations, but has problems in reaching a near optimal solution in several benchmark functions.

To overcome this problem, some researchers have employed methods with adaptive parameters [5, 6]. One is deterministic mode, which the PSO parameters are changed according to the deterministic rules, such as a linear decreased inertia weight as the number of generation increasing [5], which are obtained according to the experience. The other is adaptive mode, which adjusts the parameters according to the feedback information, such as fuzzy adaptive inertia weight [6]. However, currently, we still have not captured the relations between different parameters and their effects toward different problem instances, such as dynamic optimization problems [2], due to the complex relationship among parameters.

Unlike the former efforts on adjusting PSO parameters, this paper will propose an efficient approach by adapting the swarm on individual level, which is realized by replacing the inactive particle with a fresh one in order to maintain the social attribution of swarm, according to the analyzing for the model of PSO. Both standard and adaptive versions are compared on three benchmark problems. The results suggest that the adaptive PSO enhance the performance effectively.

2. Standard particle swarm optimization (SPSO)

PSO is similar to the other evolutionary algorithms in that the system is initialized with a population of random solutions. Each potential solution, call *particles*, flies in the D-dimensional problem space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues. The location of the ith particle is represented as $X_i = (x_{i1},\ldots,$

$x_{id}, ..., x_{iD}$). The best previous position (which giving the best fitness value) of the ith particle is recorded and represented as $P_i = (p_{i1},..., p_{id}, ..., p_{iD})$, which is also called *pbest*. The index of the best *pbest* among all the particles is represented by the symbol g. The location P_g is also called *gbest*. The velocity for the ith particle is represented as $V_i = (v_{i1},..., v_{id}, ..., v_{iD})$.

The particle swarm optimization concept consists of, at each time step, changing the velocity and location of each particle toward its *pbest* and *gbest* locations according to the equations (1a) and (1b), respectively:

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id})$$
$$\qquad + c_2 * rand() * (p_{gd} - x_{id}) \qquad (1a)$$
$$x_{id} = x_{id} + v_{id} \qquad (1b)$$

Where w is inertia weight, c_1 and c_2 are acceleration constants [2], and $rand()$ is a random function in the range [0, 1]. For equation (1a), the first part represents the inertia of pervious velocity; the second part is the "cognition" part, which represents the private thinking by itself; the third part is the "social" part, which represents the cooperation among the particles [7]. V_i is clamped to a maximum velocity $V_{max} = (v_{max,1},..., v_{max,d}, ..., v_{max,D})$. V_{max} determines the resolution with which regions between the present and the target position are searched [2].

The process for implementing PSO is as follows:

a). Set current iteration generation $G_c=1$. Initialize a population which including m particles, For the ith particle, it has random location X_i in specified space and for the dth dimension of V_i, $v_{id} = Rand_2() * v_{max,d}$, where $Rand_2()$ is a random value in the range [-1, 1];

b). Evaluate the fitness for each particle;

c). Compare the evaluated fitness value of each particle with its *pbest*. If current value is better than *pbest*, then set the current location as the *pbest* location. Furthermore, if current value is better than *gbest*, then reset *gbest* to the current index in particle array;

d). Change the velocity and location of the particle according to the equations (1a) and (1b), respectively;

e). $G_c=G_c +1$, loop to step b) until a stop criterion is met, usually a sufficiently good fitness value or G_c is achieve a predefined maximum generation G_{max}.

The parameters of PSO includes: number of particles m, inertia weight w, acceleration constants c_1 and c_2, maximum velocity V_{max}.

3. Adaptive particle swarm optimization (APSO)

As evolution goes on, the swarm might undergo an undesired process of diversity loss. Some particles become *inactively* while lost both of the global and local search capability in the next generations. For a particle, the lost of global search capability means that it will be only flying within a quite small space, which will be occurs when its location and *pbest* is close to *gbest* (if the *gbest* has not significant change) and its velocity is close to zero (for all dimensions) according to the equation (1); the lost of local search capability means that the possible flying cannot lead perceptible effect on its fitness. From the theory of self-organization [8], if the system is going to be in equilibrium, the evolution process will be stagnated. If *gbest* is located in a local optimum, then the swarm becomes premature convergence as all the particles become inactively.

To stimulate the swarm with sustainable development, the inactive particle should be replaced by a fresh one adaptively so as to keeping the non-linear relations of feedback in equation (1) efficiently by maintaining the social diversity of swarm.

However, it is hard to identify the inactive particles, since the local search capability of a particle is highly depended on the specific location in the complex fitness landscape for different problems.

Fortunately, the precision requirement for fitness value is more easily to be decided for specified problem. The adaptive PSO is executed by substituting the step d) of standard PSO process, as the pseudocode of adaptive PSO that is shown in Fig. 1. F_i is the fitness of the ith particle, F_{gbest} is the fitness of *gbest*. $\Delta F_i = f(F_i, F_{gbest})$, where $f(x)$ is a error function. The ε is a predefined *critical constant* according to the precision requirement.

T_c is the *count constant*. The *replace*() function is employed to replace the *i*th particle, where the X_i and V_i is reinitialized by following the process in step a) of standard PSO, and its *pbest* is equal to X_i.

The array *similarCount*[*i*] is employed to store the counts which is satisfying the condition $|\Delta F_i| < \varepsilon$ in successively for the *i*th particle which is not *gbest*. The inactive particle is natural to satisfy the replace condition; however, if the particle is not inactively, it has less chance to be replaced as T_c increases.

```
int[] similarCount = new int[m];   // at initialization stage
// Next code is employed to replace step d)
// in standard PSO process
FOR (i=0; i<m; i++) {              // for each particle
  IF (i!=g && |ΔFᵢ| < ε )
    THEN similarCount[i] ++;  // add 1
  ELSE   similarCount[i] = 0;     // reset
  IF (similarCount[i]> Tᶜ)        // predefined count
    THEN replace (the ith particle);
  ELSE   execute (step d) in standard PSO);
}
```

FIG. 1   Inserted pseudocode of adaptive PSO

4. Results and discussion

For comparison, three benchmark functions that are commonly used in the evolutionary computation literature [4-6] are used. All functions have same minimum value, which are equal to zero.

The function f_1 is the Rosenbrock function:

$$f_1(x) = \sum_{d=1}^{D-1}(100(x_{d+1} - x_d^2)^2 + (x_d - 1)^2) \qquad (2a)$$

The function f_2 is the generalized Rastrigin function:

$$f_2(x) = \sum_{d=1}^{D}(x_d^2 - 10\cos(2\pi x_d) + 10) \qquad (2b)$$

The function f_3 is the generalized Griewank function:

$$f_3(x) = \frac{1}{4000}\sum_{d=1}^{D} x_d^2 - \prod_{d=1}^{D}\cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \qquad (2c)$$

For the purpose of comparison, the asymmetric initialization method used in [4-6] is adopted here for population initialization. Table 1 lists the initialization ranges, and table 2 lists the V_{max} and X_{max} values for all the functions, respectively. The acceleration constants are set as: $c_1 = c_2 = 2$. The fitness value is set as function value. We had 500 trial runs for every instance.

For APSO, ΔF_i is set as a relative error function, which is $(F_i - F_{gbest})/MIN(ABS(F_i), ABS(F_{gbest}))$, where ABS(x) gets the absolute value of *x*, MIN(x_1, x_2) gets the minimum value between x_1 and x_2. The critical constant ε is set as 1e-4, and the count constant T_c is set as 3.

In order to investigate whether the adaptive PSO scales well or not, different numbers of particles *m* are used for each function which different dimensions. The numbers of particles *m* are 20, 40, 80 and 160. G_{max} is set as 1000, 1500 and 2000 generations corresponding to the dimensions 10, 20 and 30, respectively.

Table 1: Asymmetric initialization ranges

Function	Asymmetric Initialization Range
f_1	(15,30)
f_2	(2.56,5.12)
f_3	(300,600)

Table 2: V_{max} and X_{max} values for each function

Function	X_{max}	V_{max}
f_1	100	100
f_2	10	10
f_3	600	600

TABLE 3: The mean fitness values for the Rosenbrock function

m	D.	G_{max}	$SPSO_L$[5]	$FPSO$[6]	$SPSO_{0.4}$	APSO
20	10	1000	96.1715	66.01409	44.1374	28.8179
	20	1500	214.6764	108.2865	87.2810	62.0391
	30	2000	316.4468	183.8037	132.5973	95.5643
40	10	1000	70.2139	48.76523	24.3512	16.4028
	20	1500	180.9671	63.88408	47.7243	37.1774
	30	2000	299.7061	175.0093	66.6341	57.3000
80	10	1000	36.2945	15.81645	15.3883	11.6443
	20	1500	87.2802	45.99998	40.6403	32.5682
	30	2000	205.5596	124.4184	63.4453	55.2538
160	10	1000	24.4477	-	11.6283	6.9480
	20	1500	72.8190	-	28.9142	25.9617
	30	2000	131.5866	-	56.6689	45.6651

TABLE 4: The mean fitness values for the Rastrigrin function

m	D	G_{max}	$SPSO_L$[5]	$FPSO$[6]	$SPSO_{0.4}$	$APSO$
20	10	1000	5.5572	4.955165	9.9483	1.3593
	20	1500	22.8892	23.27334	44.3457	9.4018
	30	2000	47.2941	48.47555	97.0551	24.4042
40	10	1000	3.5623	3.283368	5.6853	0.4774
	20	1500	16.3504	15.04448	29.5543	4.6171
	30	2000	38.5250	35.20146	68.028	12.8490
80	10	1000	2.5379	2.328207	3.5988	0.0748
	20	1500	13.4263	10.86099	20.6500	2.3226
	30	2000	29.3063	22.52393	49.3440	7.2542
160	10	1000	1.4943	-	2.1890	0.0020
	20	1500	10.3696	-	15.3106	0.9620
	30	2000	24.0864	-	37.0096	4.0646

TABLE 5: The mean fitness values for the Griewank function

m	D.	G_{max}	$SPSO_L$[5]	$FPSO$[6]	$SPSO_{0.4}$	$APSO$
20	10	1000	0.0919	0.091623	0.09203	0.06817
	20	1500	0.0303	0.027275	0.03174	0.02589
	30	2000	0.0182	0.02156	0.0482	0.02318
40	10	1000	0.0862	0.075674	0.07617	0.05566
	20	1500	0.0286	0.031232	0.02272	0.02107
	30	2000	0.0127	0.012198	0.01527	0.01359
80	10	1000	0.0760	0.068323	0.06581	0.05258
	20	1500	0.0288	0.025956	0.02217	0.02037
	30	2000	0.0128	0.014945	0.01208	0.01049
160	10	1000	0.0628	-	0.05773	0.04344
	20	1500	0.0300	-	0.0215	0.01817
	30	2000	0.0127	-	0.01208	0.01037

Table 3 to 5 lists the mean fitness values for the three benchmark functions. Where *SPSO* is the results of standard PSO in [5] with a linearly decreasing w which from 0.9 to 0.4, *FPSO* is the results of fuzzy adaptive PSO in [6] with fuzzy adaptive w, and $SPSO_{0.4}$ is the results of standard PSO with w=0.4 as an version for comparison, *APSO* is the results of adaptive PSO in this work with w=0.4.

By compare the results, it is easy to see that *APSO* have better results than all of other PSO version for almost all cases. For example, as $SPSO_{0.4}$ get worst solutions due to premature convergence for Rastrigrin function, the adaptive version *APSO* has the capacity to sustainable development. It means that the adaptive PSO has the capacity to achieve high performance.

5. Conclusion

In this paper, an adaptive particle swarm optimizer was introduced to improve the performance. The adaptive criterion is appended on individual level. Since the critical constant ε is decided by the precision requirement to fitness, it is more easily to be decided for different problems. Three benchmark functions have been used for testing. The simulation results illustrate the performance of adaptive PSO can improve the performance. This adaptive method may be also used for other evolutionary computation technologies, such as genetic algorithms.

References

[1] J. Kennedy, R. Eberhart. Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks, 1995: 1942-1948

[2] R. Eberhart, Y. Shi. Particle swarm optimization: developments, applications and resources. IEEE Int. Conf. on Evolutionary Computation, 2001: 81-86

[3] Y. Shi, R. Eberhart. Parameter selection in particle swarm optimization. Proc. of 7th Annual Conf. on Evolutionary Programming, 1998: 591-600

[4] P. J. Angeline. Evolutionary optimization versus particle swarm optimization: philosophy and performance difference. Proc. of 7th Annual Conf. on Evolutionary Programming, 1998: 601-610

[5] Y. Shi, R. Eberhart. Empirical study of particle swarm optimization. Proc. of Congress on Evolutionary Computation, 1999: 1945-1950

[6] Y. Shi, R. Eberhart. Fuzzy adaptive particle swarm optimization. IEEE Int. Conf. on Evolutionary Computation, 2001: 101-106

[7] J. Kennedy. The particle swarm: social adaptation of knowledge. IEEE Int. Conf. on Evolutionary Computation, 1997: 303-308

[8] G. Nicolis, I. Prigogine. Self-organization in nonequilibrium systems: from dissipative systems to order through fluctuations. John Wiley, NY, 1977