Fast Multi-swarm Optimization for Dynamic Optimization Problems

Changhe Li
Department of Computer Science
University of Leicester
University Road, Leicester LE1 7RH, UK
cl160@le.ac.uk

Shengxiang Yang
Department of Computer Science
University of Leicester
University Road, Leicester LE1 7RH, UK
s.yang@mcs.le.ac.uk

Abstract

In the real world, many applications are non-stationary optimization problems. This requires that the optimization algorithms need to not only find the global optimal solution but also track the trajectory of the changing global best solution in a dynamic environment. To achieve this, this paper proposes a multi-swarm algorithm based on fast particle swarm optimization for dynamic optimization problems. The algorithm employs a mechanism to track multiple peaks by preventing overcrowding at a peak and a fast particle swarm optimization algorithm as a local search method to find the near optimal solutions in a local promising region in the search space. The moving peaks benchmark function is used to test the performance of the proposed algorithm. The numerical experimental results show the efficiency of the proposed algorithm for dynamic optimization problems.

1 Introduction

Most research in evolutionary computation focuses on static optimization problems. However, many real-world problems are dynamic optimization problems (DOPs), where changes occur over time. This requires optimization algorithms to not only find the optimal solution in a short time but also track the optimal solution in a dynamic environment. Hence, optimization methods that are capable of continuously adapting the solution to a changing environment are needed. Particle swarm optimization (PSO) is a versatile population-based stochastic optimization technique. Similar to other evolutionary algorithms (EAs) in many respects, PSO has been shown to perform well for many static problems [14], and several PSO based algorithms have been recently proposed to address DOPs [1, 4, 10, 16].

It is difficult for the standard PSO to optimize DOPs. The difficulties lie in two aspects: one is the outdated memory due to the changing environment and the other is the diversity loss due to convergence. Of these two difficulties, the diversity loss is by far more serious [4]. It has been demonstrated that the time taken for a partially converged swarm to re-diversify, find the shifted peak, and then re-converge is quite deleterious to the performance of PSO [2].

This paper introduces a multi-swarm method to maintain the diversity through the run. One parent swarm maintains the diversity and detects the promising search area in the whole search space using the fast evolutionary programming (FEP) algorithm [17], and a group of child swarms explore the local area to search for the local optima using a fast PSO (FPSO)[12] algorithm. This mechanism makes the child swarms spread out over the highest multiple peaks, as many as possible, and FPSO[12] guarantees to converge onto a local optimum in a short time.

2 Relevant Work

2.1 Particle Swarm Optimization

PSO was first introduced by Kennedy and Eberhart in 1995 [11, 8]. Each particle is represented by a position and a velocity. The velocity is updated according to the following equation:

$$\vec{V}_i' = \omega\vec{V}_i + \eta_1 r_1(\vec{P}_i - \vec{X}_i) + \eta_2 r_2(\vec{P}_g - \vec{X}_i) \quad (1)$$

where $\vec{V}_i$ and $\vec{V}_i'$ are the previous and the current velocity of particle i, $\vec{X}_i$ is the position of particle i, and $\vec{P}_i$ and $\vec{P}_g$ are the best so far position found by the individual and the best so far position found by the whole swarm respectively. ω ($\omega \in [0.0, 1.0)$) is an inertia weight that determines how much the previous velocity is preserved, η_1 and η_2 are acceleration constants, r_1 and r_2 are random numbers in the interval of [0.0, 1.0]. The position of particle i is modified by the following equation:

$$\vec{X}_i' = \vec{X}_i + \vec{V}_i', \quad (2)$$

978-0-7695-3304-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ICNC.2008.313

where $\vec{X}_i'$ and $\vec{X}_i$ represent the current and previous positions of particle i respectively.

From the theoretical analysis of the trajectory of a PSO particle [7], the trajectory of a particle $\vec{X}_i$ converges to a weighted mean of $\vec{P}_i$ and $\vec{P}_g$. Whenever the particle converges, it will "fly" to the individual best position and the global best position. According to the update equation, the individual best position of the particle will gradually move closer to the global best position. Therefore, all the particles will converge onto the global best particle's position. In fact, the particles usually converge on a local optimum.

FPSO[12] combines PSO with Cauchy mutation and evolutionary selection strategy. Experimental results show that it keeps the fast convergence speed characteristic of PSO, and greatly overcomes the tendency of trapping into local optimum of PSO.

2.2 Multiple Populations or Swarms

Many researchers have considered multi-populations as a means of enhancing the diversity of EAs to address DOPs. Branke et al. proposed a self organizing scouts (SOS) [6] algorithm that has been shown to give excellent results on the many peaks benchmark. Zeng et al. proposed an orthogonal design based evolutionary algorithm, called ODEA, where its population consists of "niches" and an orthogonal design method is employed. ODEA borrows some ideas from the SOS algorithm, however, the experimental results show that the performance of ODEA is better than the SOS algorithm.

Parrott and Li developed a speciation based PSO (SPSO) [13], which dynamically adjusts the number and size of swarms by constructing an ordered list of particles, ranked according to their fitness, with spatially close particles joining a particular species.The atomic swarm approach has been adapted to track multiple optima simultaneously with multiple swarms in dynamic environment by Blackwell and Branke [3, 4]. In their approach, a charged swarm is used for maintaining the diversity of the swarm, and the exclusion principle ensures that no more than one swarm surrounds a single peak. This strategy is very efficient for the moving peaks benchmark (MPB) function [5].

3 Fast Multi-Swarm Optimization

The multi-population idea is particular useful in multimodal environments, such as many peaks. Here, using multi-swarm is to divide the whole search space into many sub-spaces, each swarm converging on a promising peak in a local environment. In FMSO, a parent swarm as a basic swarm is used to detect the most promising area when the environment changes, and a group of child swarms are used to search the local optimum in their own sub-spaces. Here, we adopt the idea of radius in SPSO [13]. Each child swarm has its own search region defined as a sphere of radius r and centers on its best particle $\vec{s}$. Hence, a particle $\vec{x}$ with its distance less than r from $\vec{s}$ belongs to the child swarm. The distance $d(\vec{x}, \vec{s})$ between two points $\vec{x}$ and $\vec{s}$ in the n-dimension space is defined as the Euclidean distance as follows:

$$d(\vec{x}, \vec{s}) = \sqrt{\sum_{i=1}^{n}(x_i - s_i)^2} \quad (3)$$

According to our experimental experience, the more peaks in the landscape, the more child swarms are relatively needed. r is relative to the range of the landscape and the width of peaks. In general, we set r according to the following equation:

$$r = \sqrt{\sum_{i=1}^{n}(x_i^u - x_i^l)}/(W_{min} + c(W_{max} - W_{min})), \quad (4)$$

where x_i^l and x_i^u are the lower and upper bound on the i-th dimension of the variable vector of n dimensions. W_{min} and W_{max} are the minimum and maximum peak width, c is a constant number in the range of $(0, 1)$. Although this method assumes that the minimum and maximum peak widths are known, it is useful to MPB problems.

Child swarms will not overcrowd on a single peak because of no overlap among child swarms. On the contrary, they will spread out over as many different peaks as possible. In FMSO, two search algorithms are used in the parent swarm and child swarms respectively. The FEP algorithm [17] has a good global search capability because of its higher probability of making longer jumps. Using this algorithm as a global search operator in the parent swarm is favorable for maintaining diversity and detecting the most promising search area. Fast convergence ratio of the FPSO algorithm makes it suitable to be a local search operator.

FMSO starts with a single parent swarm searching through the entire search space. With the iteration, if the best particle of the parent swarm gets better, then a new child swarm is generated around the center of the best particle with a radius r. If the distance between a particle in the parent swarm and the center of the child swarm is less than r, then the particle of the parent swarm is moved to the child swarm and, correspondingly, a new particle is randomly generated into the parent swarm. That is, if the best particle in the parent swarm gets better, it means that a promising search area may be found. Then a child swarm is split off from the parent swarm and a sub-search region is generated. Because of no overlap among child swarms, if the distance between two child swarms is less then their radius, then remove the whole swarm of the worse one. This guarantees that no more than one child swarm will spread over a single peak.

In FMSO, each child swarm has a failure counter which monitors the improvement of the fitness value for the best particle of each child swarm. If there is no improvement of the fitness, then the failure counter is increased by one in each iteration until this value achieves a pre-specified value, named $maximum\_failure\_counter$. When the failure counter becomes equal to the $maximum\_failure\_counter$, we use a mutation operator for the best particle to make it jump to a new position. The mutation operator for the best particle is as follows:

$$\vec{X}' = \vec{X} + \eta N(0,1), \tag{5}$$

where η denotes a scale parameter and $N(0,1)$ is a random number generated according to a Gaussian distribution.

The number of child swarms will increase with the best particle being improved in the parent swarm. However, it will not surpass a maximum number. When the number of child swarms increases to the maximum number of child swarms and if the best particle is improved in the parent swarm at this moment, then the child swarm with the biggest failure counter is replaced by a new child swarm. Generally speaking, the more peaks there are in the landscape, the bigger the maximum number of child swarms will be. However, the size of child swarms is very small, usually set to 5-10.

The major steps of FMSO are summarized as follows:

Step 1: Randomly initialize the parent swarm

Step 2: Evolve the parent swarm using the FEP operator

Step 3: If the best particle of the parent swarm turns better, produce a child swarm around the best one (center of the child swarm) with a radius r. If the number of child swarms reaches the maximum number of child swarms, replace the child swarm with the biggest failure counter by the new one

Step 4: Update the parent swarm. If any particle of the parent swarm is within the earch area of a child swarm, then move the particle into the child swarm and randomly generate a new particle in the parent swarm

Step 5: Evolve the child swarms with the FPSO operator. If the $best[i]$ of swarm i gets better, $failure\_counter[i] = 0$; else, $failure\_counter[i] = failure\_counter[i] + 1$.

Step 6: For each child swarm i, if $failure_counter[i] = maximum\_failure\_counter$, the mutation operator is called

Step 7: Update each child swarm. If one child swarm is within the search radius of another child swarm, destroy the worse one

Step 8: If the termination criterion is satisfied, then stop; otherwise, goto Step 2.

4 Experimental Study

4.1 Dynamic Test Function

Branke [5] suggested a dynamic benchmark problem, called "moving peaks" benchmark (MPB) problem. It consists of a multi-dimensional landscape with several peaks, where the height, width and position of each peak is altered a little every time an environmental change occurs. We choose this dynamic function to construct our test problems. This function is capable of generating a given number of peaks in a given number of dimensions that vary both spatially (position and shape of the peak) and in terms of fitness. More details of the MPB function can be found at the website: http://www.aifb.uni-karlsruhe.de/∼jbr/MovPeaks/.

The default settings and definion of the benchmark used in the experiments can be found in Table 1. The term "evaluation" means that an individual is created and its fitness is evaluated. Shift length s means that a peak p will move within its neighborhood with a radius s after the next environmental change. f denotes the change frequency, A denotes the minimum and maximum allele values, H denotes the minimum and maximum heights of the peaks, W denotes the minimum and maximum peak width parameters, and I denotes the initial peak heights for all peaks.

Table 1. Default settings for the moving peaks benchmark

Parameter	Value
m (number of peaks)	10
f	every 5000 evaluations
height severity	7.0
width severity	1.0
peak shape	cone
basic function	no
shift length s	1.0
number of dimensions	5
A	[0, 100]
H	[30.0, 70.0]
W	[1, 12]
I	50.0

4.2 Experimental Settings

In this study we compare the performance of FMSO with the ODEA algorithm [18]. For FMSO, based on the previous experimental work by den Bergh [15], the acceleration constants η_1 and η_2 were both set to 1.496180, and the inertia weight $\omega = 0.729844$. The default maximum number

of child swarms was set to 10 and the default value of r was set to 25.0. The size of parent and child swarm was set to 100 and 10 for all the experiments respectively. In order to fairly compare FMSO with ODEA, the parameter setting for FMSO is the same as ODEA. Since each particle in FMSO is evaluated twice at each generation, the evaluation number for each particle increases by 2.

For evaluating the efficiency of the algorithms, we use the offline performance measure as follows:

$$e_t = \frac{h_t}{f_t}, \tag{6}$$

where f_t is the best solution got by FMSO just before the t-th environmental change, h_t is the optimum value at time t, and e_t is the relative value of f_t and h_t. The offline error of an algorithm is defined as:

$$\mu = \frac{1}{T}\sum_{t=1}^{T}(h_t - f_t), \tag{7}$$

where μ is the average of all differences between h_t and f_t over the environmental changes and T is the number of environmental changes.

4.3 Experimental Results

For all results reported here, we use an average over 50 runs (environment changes 50 times) of the FMSO algorithm. The offline errors after 5,000 evaluations in the FMSO algorithm are compared with those of the ODEA algorithm. Tables 2, 3, and 4 show the effect of peak movements, frequency of change, and the number of peaks, on the performance of both the FMSO and ODEA algorithms respectively. Table 5 shows the results of varying the radius r of child swarms by the FMSO algorithm.

Table 2. The offline errors (μ) of FMSO and ODEA for different shift lengths.

shift length	FMSO	ODEA
1.0	0.090467	1.95
2.0	0.120339	2.23
3.0	0.180055	2.38

The offline error of FMSO is much smaller than that of ODEA in Table 2. By observing Table 2, we can easily see that FMSO is more suitable to the largely dynamic environment (the position, height, and shape of the highest peak change greatly between two runs) than ODEA. All the results show that FMSO can track the moving best solution more accurately than the ODEA algorithm.

Table 3. The offline errors (μ) of FMSO and ODEA for different number of individual evaluations between changes.

evaluations	FMSO	ODEA
10000	0.0106228	1.81
5000	0.0904674	1.95
2500	0.393887	2.19
1000	1.16768	2.43
500	2.5733	4.70

Table 4. The offline errors (μ) of FMSO and ODEA for different number of peaks, where C_{max} is the maximum number of child swarms.

number of peaks	C_{max}	FMSO	ODEA
1	5	0.00437849	0.68
10	10	0.183839	1.95
20	10	0.509487	1.93
30	10	0.69417	1.91
40	20	0.936065	1.82
50	20	0.937797	1.70
100	20	0.909333	1.31
200	30	1.18632	1.14

Table 5. The offline errors (μ) of FMSO for different radius r of child swarms.

r	5.0	15.0	25.0	35.0	40.0
μ	1.431	0.984	0.152	0.810	1.930

By observing the offline performance in Figs. 1 to 3, we can easily see that the offline performance e_t is nearly equal to 1 for most t. That is, FMSO is capable of tracking the highest peaks when the environment changes.

Table 5 shows the sensitivity analysis results regarding the radius of child swarms for the performance of FMSO. We use radius r to control the search region of each child swarm. Therefore, it plays a critical role in the performance of FMSO. If r is too small,there is a potential problem that the small isolated child swarms will converge on a local optimum, the diversity will lose and it hardly makes progress. However, if r is too large, there will be more than one peaks within the search region of a child swarm, and the number of child swarms will decrease. When r is large enough to cover the whole search space, FMSO degenerates to a stan-

dard PSO, which just uses a single swarm to locate a global optimum. We test the performance of FMSO with r ranging from 5.0 to 40.0 (we can adjust the constant number c in Eq. (4) to get the corresponding r). By observing Table 5, the offline errors of both too small r values and too large r values are relatively poor. When the value of r increases form 5.0 to 25.0, the offline error turn to be the smallest. When r further increases to 40.0, the offline error gets lager again. Hence, the performance of FMSO is much sensitive to the radius of child swarms.

5 Conclusions

This paper proposes a fast multi-swarm optimization algorithm (FMSO) for dynamic optimization problems. Two type of swarms are used in FMSO: one is the parent swarm to detect the promising area in the whole search space and the other is child swarms to explore the local optimum in a local area found by the parent swarm. By using FEP and FPSO operators in the parent swarm and child swarms respectively, it can make the child swarms spread out over the highest peaks and converge on the global optimum. The experimental results show that FMSO can not only find the global or near global optimum, but also track the moving best solution in a dynamic environment.

References

[1] T. M. Blackwell. Swarms in dynamic environments. *Proc. of the 2003 Genetic and Evolutionary Computation Conference*, LNCS 2723, pp. 1-12, 2003.

[2] T. M. Blackwel. Particle swarms and population diversity II: Experiments. *Proc. of the 2003 Genetic and Evol. Comput. Workshops*, pp. 108-112, 2003.

[3] T. M. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In G. R. Raidl, editor, *Applications of Evolutionary Computing*, LNCS 3005, pp. 489-500, 2004.

[4] T. M. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, **10**(4): 459-472, 2006.

[5] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. *Proc. of the 1999 Congr. on Evol. Comput.*, vol. 3, pp. 1875-1882, 1999.

[6] J. Branke, T. Kaußler, C. Schmidth, and H. Schmeck. A multi-population approach to dynamic optimization problems. *Proc. 4th Int. Conf. on Adaptive Computing in Design and Manufacturing*, pp. 299-308, 2000.

[7] M. Clerc and J. Kennedy. The particle swarm: Explosion, stability and convergence in a multi-dimensional complex space. *IEEE Trans. on Evolutionary Computation*, **6**: 58-73, 2002.

[8] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory, *Proc. of the 6th Int. Symp. on Micro Machine and Human Science*, pp. 39-43, 1995.

[9] D. Fogel. An introduction to simulated evolutionary optimization. *IEEE Trans. on Neural Networks*, **5**(1): 3-14, 1994.

[10] S. Janson and M. Middendorf. A hierachical particle swarm optimizer for dynamic optimization problems. In G. R. Raidl, editor, *Applications of Evolutionary Computing*, LNCS 3005, pp. 513-524, 2004.

[11] J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. *Proc. of the 1995 IEEE Int. Conf. on Neural Networks*, pp. 1942-1948, 1995.

[12] C. Li, Y. Liu, L. Kang and AM. Zhou,A Fast Particle Swarm Optimization Algorithm with Cauchy-Mutation and Natural Selection Strategy. *Springer Press,ISICA2007, LNCS4683*, pp.334 - 343.

[13] D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. *Proc. of the 2004 IEEE Congress on Evolutionary Computation*, pp. 98-103, 2004.

[14] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, **1**(2-3): 235-306, 2002.

[15] F. van den Bergh. An analysis of particle swarm optimizers. *PhD Thesis*, Department of Computer Science, University of Pretoria, South Africa, 2002.

[16] H. Wang, D. Wang, and S. Yang. Triggered memory-based swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, LNCS 4448, pp. 637-646, 2007.

[17] X. Yao and Y. Liu. Fast evolutionary programming. *Proc. of the 5th Annual Conference on Evolutionary Programming*, pp. 451-460, 1996.

[18] S. Zeng, H. de Garis, J. He, and L. Kang. A novel evolutionary algorithm based on an orthogonal design for dynamic optimization problems. *Proc. of the 2005 IEEE Congress on Evol. Comput.*, vol. 2, pp. 1188 - 1195, 2005.