ATLANTIS PRESS

Advances in Engineering Research, volume 113

International Forum on Mechanical, Control and Automation (IFMCA 2016)

Multi-swarm hybrid optimization algorithm with prediction strategy for dynamic optimization problems

Wenbo nie[1,a], Lihong Xu[2,b]

[1,2] College of Electronics and Information Engineering

Tongji University,Shanghai, China

[a]wenbonie83@163.com, [b]xulhk@163.com

Keywords: Dynamic optimization algorithm. Particle swarm optimization. Simulated Annealing. Prediction strategy

Abstract. It is known that optimization in a changing environment is a challenging task, for which the basic goal is not only to obtain the optimal solution, but also strongly adapting to the environmental changes and tracking the optimal solution as closely as possible. In this paper, a novel multi-swarm optimization algorithm is proposed for solving dynamic optimization problems (DOPs) effectively, which is based on the hybrid of particle swarm optimization (PSO) and Simulated Annealing (SA) with an prediction strategy. Firstly, an multi-swarm strategy is adopted, which simultaneously employs PSO method to conduct global search for exploring promising optimal solutions and adopt SA to conduct local search. Secondly, a new forecasting model is developed by using the principle that the previous optimum locations can predict the optimum's location in the changing environment, which can improve the performance of the algorithm in dynamic environment. Then, a diversity preservation mechanism is incorporated into our method to obtain more robust results. Experiments are conducted on the set of benchmark functions used in CEC 2009 competition for DOPs, and the results show that the proposed algorithm achieves good performance and outperforms others in solving DOPs with the model changed by following some pattern.

Introduction

In the real-world, many optimization problems are changing over time. Examples include scheduling where new jobs arrive over time, or the vehicle routing with new requests arriving over time. In the problem of greenhouse control, the influence of outside and inside environment (humidity, temperature, CO2, illumination, etc.) on plant growth always changes over time. Recently, dynamic optimization problems (DOPs) have raised great attention and interest for its significance in real-world applications [1]. When coping with DOPs, the goal of the optimization algorithms not only focuses on obtaining the optimal solution, but also to have strong adaptive capability to the environmental changes and track the trajectory of the optimal solution as closely as possible [2].

Particle swarm optimization (PSO) [3] and Simulated Annealing (SA) [4] are very popular optimization algorithms. They have drawn much attention of many researchers because of its excellent performances for continuous optimization problems.

In this paper, inspired by the multi-swarm algorithm proposed by Tim Blackwell and Branke, also based on part of our laboratory's achievement [5]. To improve the performance of the algorithm, a new multi-swarm optimization method with prediction strategy based on PSO and SA has been proposed. One explorer swarm detects the promising peaks in the search space, and a number of local swarms perform local search to find the local optimums and track them. In addition, the prediction strategy with diversity increasing mechanism is proposed to improve the accuracy of the algorithm. the typical dynamic rotation peak benchmark generator is adopted. The results show that Multi-SAPSO-PRE has very well performance on a variety of cases of test cases especially for the regular change pattern.

Copyright © 2017, the Authors. Published by Atlantis Press.
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

The rest of this paper is organized as follows. Sect. 2 introduces the Multi-SAPSO-PRE algorithm in detail. Experiments and comparison results are given in Section 3. Finally, the conclusion is drawn in the last section.

Mutil-SAPSO With Prediction Strategy Algorithm

Particle Swarm Optimization (PSO). Particle Swarm Optimization, a well known swarm intelligent optimization algorithms, was first introduced by Kennedy and Eberhart in 1995 [3]. To Simulate the behaviors of flock foraging, particles in PSO move through the solution space to search for the global optimum by mutual cooperation and sharing information with each other. In this paper we simultaneously utilize PSO to conduct global search.

Assuming D is the dimension of the search space and N is the population size. For each particle *i*, the current position vector is $\vec{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$, and the velocity vector is $\vec{v}_i = (v_{i1}, v_{i2}, ..., v_{iD})$, where $1 \le i \le N$. A vector $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$ denotes the personal best position of particle i found so far by itself and a vector $G = (g_1, g_2, ..., g_D)$ denotes the position of the global best solution found so far by the whole swarm. At the beginning of the algorithm, the positions and the velocities of all particles are initialized randomly. Then, during the iteration process, the velocity and the position of each particle are updated respectively according to Eq. 2 and Eq. 3.

$$v_{id}(t+1) = c[v_{id}(t) + c_1 r_{1d}(P_{id}(t) - x_{id}(t)) + c_2 r_{2d}(G_d(t) - x_{id}(t))] \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

where c is the constriction factor, c_1 and c_2 are two acceleration coefficients, r_1 and r_2 are two random numbers obeying a uniform distribution in [0, 1].

Accordingly, the procedures of the PSO algorithm can be summarized as follows:

Step 1: Randomly initialize positions and velocities of all particles in the search space.

Step 2: For each particle *i*, set P_i to be its current position, then evaluate $fitness(P_i)$.

Step 3: Set $G = \arg\max\{fitness(P_i)\}$.

Step 4: Apply Eqs. (1) and (2) to update the velocity and position of each particle.

Step 5: Evaluate $fitness(\vec{x}_i)$, if $fitness(\vec{x}_i) > fitness(P_i)$, $P_i = \vec{x}_i$; if $fitness(P_i) > fitness(G)$, $G = P_i$.

Step 6: Repeat the Step 4 to Step 5 until the termination criterion reached.

Simulated Annealing (SA). Simulated annealing (SA) is a random search algorithm based on the thermodynamic annealing mechanism [4]. To begin with an initial solution and the initial temperature value (also called control parameter) T, SA generates a new solution in its neighborhood. If the quality of the new solution is higher than the original one, then this solution is received; on the contrary, SA receives the new solution with a certain probability decreased with time. Therefore,When the algorithm terminates, the current solution obtained is the approximate optimal solution.

It is known that Simulated Annealing has strong local search ability, as well as preventing the swarm from falling into local optima. What's more, its capability of accepting bad solutions with a certain probability can figure out the situation mentioned above. Algorithm 1 shows the pseudo-code for SA local search in the proposed algorithm.

Algorithm 1 : *Pseudo-code for SA local search ( )*

```
1.  Setting parameters T , T0 , L , a
2.    for each local swarm i do
3.      iter _ num = 1
4.      while ( iter _ num < L ) do
5.        for each partile j do
6.          update the velocity of partile j according to Eq.(1)
7.          update the position of partile j according to Eq.(2)
8.          if ( fitness( x_j ) > fitness( P_j )) then
9.            P_j = x_j
10.         else
11.           apply Metropolis criteria
12.           if accepted then
13.             P_j = x_j
14.           end if
15.         end if
16.       end for
17.       iter _ num = iter _ num + 1
18.     end while
19.     T = a * T
20.     update G_i
21.   end for
```

In the local search procedure, namely Algorithm 1, T_0 is the initial temperature, L is the Markov chain length and a is the temperature decline coefficient. At the beginning, a new position is chosen for particle j in local swarm i. The fitness value of the new position $fitness(\mathbf{r}_j)$ is then computed. If $fitness(\mathbf{r}_j)$ is greater than $fitness(P_j)$, the new position is accepted as the personal best position of particle j. If $fitness(\mathbf{r}_j)$ is less than or equal to $fitness(P_j)$, the metropolis criteria decide on acceptance [4]. The probability value

$$p = e^{(fitness(\mathbf{r}_j) - fitness(P_j))/T} \tag{3}$$

is computed and compared to a random number uniformly distributed in (0, 1). If p is greater than the random number, the new position is accepted. Otherwise, the new position is rejected. After L times through the above process, the value of temperature T is decreased according to the temperature decline function

$$T = a * T \tag{4}$$

where a is in the range of (0, 1). A lower temperature makes a bad solution to be accepted with a less probability.

Multi-swarm Mechanism. Multi-swarm approach is particularly useful in multi-modal environments [6]. Here, we use multi-swarm mechanism to cover and track different promising peaks in the whole search space. In the proposed algorithm, there is one global swarm and a number of sub-swarms. The global swarm is responsible to explore the entire search space to find promising

peaks, while sub-swarms are used to cover different peaks and perform local search to find the tops of the peaks they covered.

At the beginning, the global search is conducted. During the iteration process, the global swarm generates a sub-swarm to cover a peak when converging to the peak. When the global swarm converges, the better particles in it are split to generate a sub-swarm, and their parameters are copied into the sub-swarm at the same time. The sub-swarm will be applicable for finding the top of the peak and tracking that peak as closely as possible if the environment changes. Then the global swarm begins detecting the search space for the sake of finding new promising peaks not already covered. The above procedure will repeat insistently until all the peaks are found by the global swarm and covered by different sub-swarms respectively.

Prediction Strategy. For dynamic optimization problems, when the environment changes, how to track changes in the environment has become challenging in designing algorithms.
Forecast mechanism, which uses the past history of the optimum's path over time to predict the optimum's location in the next environmental change time is an effective way to cope with the dynamic optimization problem. The approach proposed in this paper makes use of the past history of the optimum's path over time to predict the optimum's location in the next environmental change time. More specifically, the past sequence of locations of the global optimum in the environmental change process can be seen as a time series. Then, using this forecast, an anticipatory group of individuals is placed on and near the estimated location of the next optimum. This prediction solutions are inserted into the population when a change in the objective landscape arrives, aiming at accelerating the convergence to the new global optimum and improve the learning ability of the algorithm. Moreover, the method is combined with some diversity preservation mechanism in the case when the forecast is unsuccessful. The followings are the details of various steps of the algorithm, as well as the specific process.

Prediction Model. Suppose that $x_{t-1}, x_{t-2}, ..., x_{t-i}, i = 1, ...t$ are a series of points (D-dimensional vectors) in the search space that describes the movements of the global optimum of the particles during the process when environment changes, a generic model to predict the location of the next global optimum can be formulated as follows and showed in Fig 1 :

$$x_{t+1} = F(x_t, x_{t-1}, ...x_{t-r+1}, t) \tag{5}$$

where *r* represents the number of the previous environmental changes, on which x_{t+1} is dependent in the prediction model. Any time series models [7] can be used for modeling *F* in (5), In the current implementation, an autoregressive (AR) model created by Schneider and Neumaier [8] which is a kind of stochastic time series models, is used as a forecasting method.

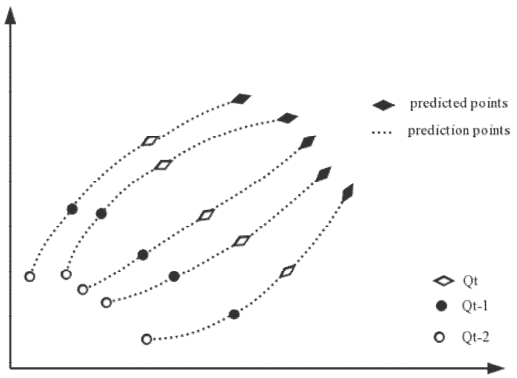

Fig 1. Schematically Forecasting Model when r=3

The Increasing Diversity Mechanism. In this algorithm, the use of prediction optimums is not meant to be completely self-sufficient for dynamic optimization. This is because there might not be predictability in the dynamic behavior of the objective function, or the pattern might not be identifiable by the forecasting method. In such a case, other dynamic optimization mechanisms are required.

After an environment change has been detected, the mechanism for increasing diversity is executed in order to address the problem of local swarms' diversity loss and in the case when the forecast is unsuccessful.

On this basis, we use a novel and simple approach to execute a small movement in a random direction to make the position of each particle deviate from its original location. What's more, owing to the velocity of each particle being almost zero before the environmental change, it also should be reset with a small random value.

The major steps are described as follows:

Step 1: Calculate the new position of each particle j in the local swarm i according to

$$\vec{x}_{i,j} = \vec{x}_{i,j} + (2 \times Rand - 1) \times (\vec{x}_{i,j} - Gbest_i) \tag{6}$$

where $Gbest_i$ represents the position of the best particle of local swarm i.

Step 2: Reset the velocity of particle j in the local swarm i according to

$$\vec{v}_{i,j} = 2 \times Rand^D - 1. \tag{7}$$

In the aforementioned steps, $Rand$ function in Eq. (6) is to generate a random number in the range of (0, 1). $Rand^D$ is a D-dimension vector of random numbers with a uniform distribution in the range of (0, 1), so $(2 \times rand^D - 1)$ is D-dimension vector in the range of $(-1, 1)^D$.

The Overall Algorithmic Process. When detecting the changes in the environment, the local swarm will be responsible for finding the top of the peak and tracking that peak as closely as possible. The total population of the local swarm at the beginning of a time step is composed of two parts: one part is generated by the prediction mechanism (called the prediction solutions), which handles the predictable portion of the objective's change pattern, and another part is generated by the increasing diversity mechanism (called the diversity solutions), in which each solution is able to handle any unpredictable change and help discover the new optimum even if the prediction remains a large error. The Overall Algorithmic Process is showed in Fig 2：

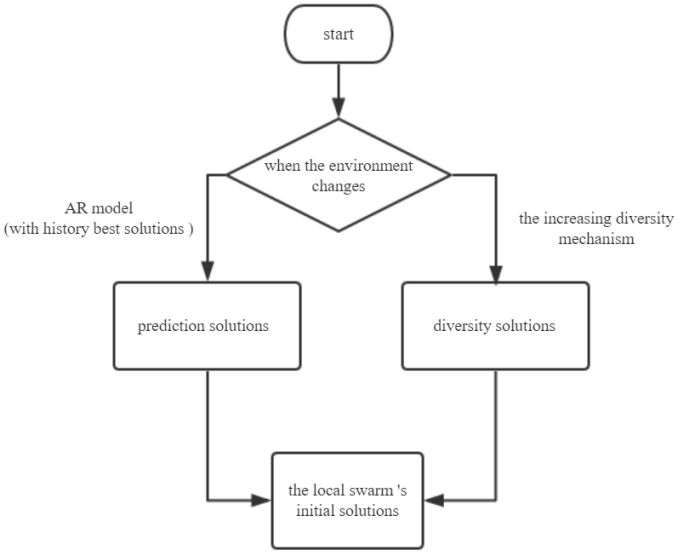

Fig 2. The Overall Algorithmic Process

The prediction strategy is outlined in the following:

At the end of the environmental change process t-1:

Step 1: In the previous environment change process, recording the history global particle of optimal solutions{ $x_t, x_{t-1}, \cdots$ }

Step 2: Using the time series of the current and the history global particle of optimal solutions and the AR forecasting method, create a prediction for the location of the next optimal solution x_{t+1} .

Step 3: A group of individuals (prediction solutions) is created by using the prediction model, update part of the population of particles' position by inserting the prediction location.

Step 4: the mechanism for increasing diversity is executed to update another part of the population of particles' position

Experimental Study

Dynamic rotation peak benchmark generator. In order to evaluate the performance of Multi-SAPSO-PRE with prediction strategy, the dynamic rotation peak benchmark generator (DRPBG) is adopted in this paper, which is one of the most widely-used benchmark for testing dynamic optimization algorithms, IEEE CEC 2009 benchmark problems for dynamic optimization was proposed by Li and Yang and mentioned this benchmark instance [9].

DRPBG has a peak-composition structure similar to those of MPB[10]; however it uses a rotation method instead of shifting the positions of peaks. The fitness function of DRPBG with n dimensions and m peaks is defined by:

$$F(\vec{x},t) = \max_{i=1,\ldots,m}\left(\frac{H_i(t)}{1+W_i(t)\sum_{j=1}^{D}\frac{(x_j - X_{ij}(t))^2}{n}}\right) \tag{8}$$

where D is the number of dimensions, m is the number of peaks, $X_{ij}(t)$ is the coordinate of the ith peak on the j-th dimension at time t, $\vec{X}_i(t)$ is the location of the ith peak, $\vec{H}$ and $\vec{W}$ are respectively the height and the width of each peak with the formulations by t as following:

$$H(t+1) = H(t) \oplus \Delta H \tag{9}$$

$$W(t+1) = W(t) \oplus \Delta W \tag{10}$$

where ΔW and ΔH are determined by different change types, for instance, those of system control parameters, like small step change (T1), large step change (T2), random change (T3), chaotic change (T4), recurrent change (T5), recurrent change with noise (T6), and random change type with changed dimension (T7).

The initial parameter settings of DRPBG used in the experiments are summarized in Table 1.

Table 1. PARAMETERS OF DRPBG

Parameter	*Value*
Number of peaks, *m*	10 or 50
Change frequency, *f*	Every 5000 evaluations
Number of dimensions, *D*	10
Peaks location range	$[-5,5]^D$
Peak height	[10, 100]
Peak width	[1, 10]

Performance metrics. Simulation environment used for carrying out the experiments described in the subsequent sections can be summarized as: CPU: 2.2 GHz Intel Core i7, RAM: 8 GB DDR3, and all methods considered in this paper are implemented using MATLAB 2014b. In order to evaluate the performance of the algorithms, the mean error [9] and the standard deviation (STD)[9] computed over 20 independent runs are utilized. which are formulated as the following:

$$E_{mean} = \frac{1}{(runs * num\_change)} \sum_{i=1}^{runs} \sum_{j=1}^{num\_change} E_{i,j}^{last} \tag{11}$$

$$STD = \sqrt{\frac{\sum_{i=1}^{runs} \sum_{j=1}^{num\_change} (E_{i,j}^{last} - E_{mean})}{runs * num\_change - 1}} \tag{12}$$

where *Gbest*(*t*) is the optimal solution found by the proposed algorithm at the t-th status of the environment, and *global optimum*(*t*) is the true global optimum in the landscape at the t-th status of the environment. It is evident that the smaller the value of the mean error and STD, the better performance the algorithm can achieve.

Results and discussion. This set of experiments is carried out to investigate the performance of Multi-SAPSO-PRE, Multi-SAPSO (our proposed algorithm without the prediction) and five other algorithms using DRPBG. In DRPBG, there are two tests, one using 10 peaks and the other using 50 peaks. The performance of our algorithm is evaluated in terms of mean and standard deviation (STD) of error values along with DASA[11], jDE[12], CPSO[13], CESO[14], PSO-CP[14]. The initial configuration values of Multi-SAPSO and Multi-SAPSO-PRE are given in Table 2.

Table 2. CONFIGURATION VALUES OF MULTI-SAPSO AND MULTI-SAPSO-PRE

Parameter	***Value***
c_1, c_2	2.05
C	0.729843788
r_c	1
X	100
the populaiton size of *explorer*	15
swarm	
the population size of *local swarm*	30
the number of “sentry” particles	5

Table 3 presents the mean error and the standard deviation of all the involved algorithms for 10 peaks. From the results, on function DRPBG with change types of small step (T1), large step (T2), recurrent (T5), recurrent with noise (T6), Multi-SAPSO-PRE remained the best average mean, indicating a better tracking of the changing optima by the proposed algorithm. And the results show the prediction strategy to have a positive effect, which is stronger when using the AR model. This is because Multi-SAPSO-PRE adopts the prediction strategy and it is more easy to track the global optimum when the objective function follows some pattern. Also it may be observed that Multi-SAPSO-PRE yielded the third best mean error in this test instance of change type random (T3) and chaotic change (T4). This is because the change pattern is random and the value of the past information may be useless, thus the prediction strategy plays with less importance when the environment changes, but the mechanism for increasing diversity contributes to the algorithm for finding the optimal solution.

As can be seen in Table 4, it can be concluded that Multi-SAPSO-PRE can obtain the best average mean for most types of changes, including small step (T1), large step (T2), recurrent (T5) and recurrent with noise (T6), and the prediction strategy can achieve an obvious beneficial effect. Also, in terms of the average mean value, we can observe that Multi-SAPSO-PRE remains the fourth one in change type random (T3) and the second one in change type chaotic change (T4), exhibiting a worse adaptability value in these two cases.

As the number of peaks increases, the problem environment becomes more complicated. We can observe that better results are obtained by Multi-SAPSO-PRE, for which the increase of the number of peaks has only a relatively small impact on mean error results. In general, among all the algorithms, Multi-SAPSO-PRE involves a much better performance under the condition of different number of peaks.

Thus, the above discussion indicates that the Multi-SAPSO-PRE is a conceptually effective approach to address time-changing problems with evolutionary algorithms, since it adopts a prediction strategy in which the optimization algorithm exploits the past information and prepares for the change before it arrives. Hence, the proposed algorithm can be a good alternative to state-of-the-art algorithms in solving the dynamic optimization problems with some regular pattern of changes.

Table 3. MEAN ERROR VALUES AND STANDARD DEVIATIONS ACHIEVED BY MULTI-SAPSO-PRE AND OTHER ALGORITHMS

Peaks	Algorithm	Error	T1	T2	T3	T4	T5	T6
10	Multi-SAPSO-PRE	Average	**0.0318**	**0.0537**	**4.5412**	**0.0515**	**1.0230**	**1.0193**
		STD	**0.0041**	**0.0307**	**8.9274**	**2.8416**	**0.1983**	**0.6432**
	Multi-SAPSO	Average	0.0636	1.4371	4.3903	0.0403	1.4782	1.0721
		STD	0.0103	0.0642	8.9750	3.4395	2.3466	2.1078
	DASA	Average	0.1912	4.2346	6.5125	0.5143	2.2799	2.8999
		STD	1.4897	8.9839	9.8784	2.7889	6.9894	9.8994
	jDE	Average	0.0402	3.7228	3.2178	0.0334	2.3664	1.4114
		STD	1.8748	8.9084	8.9849	0.9839	6.9784	7.8738
	CPSO	Average	0.0495	2.4892	4.7644	0.0712	1.4044	1.3598
		STD	0.8743	4.9823	8.8738	0.6287	8.7342	7.9783
	CESO	Average	0.0779	2.4743	5.6436	0.0988	1.6267	1.0384
		STD	2.8738	7.9823	9.9823	0.8327	5.7362	6.9278
	PSO-CP	Average	0.0424	2.7912	4.6943	0.0528	1.6781	1.5971
		STD	0.7673	8.8738	9.9283	0.9327	8.7362	7.8927

Table 4. MEAN ERROR VALUES AND STANDARD DEVIATIONS ACHIEVED BY MULTI-PSO-PRE AND OTHER ALGORITHMS

Peaks	Algorithm	Error	T1	T2	T3	T4	T5	T6
50	Multi-SAPSO-PRE	Average	**0.0592**	**0.1220**	**5.3728**	**0.1304**	**0.9844**	**0.8921**
		STD	**0.0132**	**0.0714**	**8.9561**	**2.9325**	**0.5732**	**1.5743**
	Multi-SAPSO	Average	0.1280	0.6743	4.5710	0.1022	0.9996	1.2141
		STD	0.0745	0.1651	8.0842	2.6527	2.4930	2.4572
	DASA	Average	0.3914	3.9044	7.3768	0.3498	1.1117	3.0211
		STD	2.8373	7.9308	8.9873	2.8733	4.7833	4.9833
	jDE	Average	0.1967	4.2788	4.5945	0.0975	1.0028	1.8942
		STD	0.8378	8.8736	7.8937	0.3424	3.8974	7.0922
	CPSO	Average	0.3932	4.1987	4.8375	0.3875	2.1139	1.6876

		STD	1.9837	8.7833	7.7836	0.7468	1.8739	7.7938
	CESO	Average	0.4673	4.7804	6.9167	0.1895	1.0870	0.9968
		STD	2.8733	6.8733	8.7893	1.8736	3.6732	4.8732
	PSO-CP	Average	0.9748	3.4748	4.0017	0.1134	1.1301	0.9874
		STD	0.8972	6.8793	7.8732	0.8937	3.7832	4.2982

Conclusions

In this paper, a new algorithm named as Multi-SAPSO-PRE is proposed for dynamic optimization problems. In the proposed algorithm, swarms are divided into two categories: one is the explorer swarm and the other is the exploitation swarm, aiming at covering different peaks as well as performing local search to find the tops of the peaks they covered, after then tracking them as closely as possible when the environment changes. At last, a prediction strategy is proposed, which makes use of any predictability present in the objective's behavior to improve the performance of a dynamic evolutionary algorithm. And the diversity preservation mechanism is executed to solve the problem such as the diversity loss of local swarms and in the case when the forecast is unsuccessful.

The performance of Multi-SAPSO-PRE has been tested on the CEC 2009 benchmark on DOP's of DRPBG and the results have been found to be far superior to that obtained from several state-of-the-art algorithms. The experimental results show that Multi-SAPSO-PRE has good performance when the dynamic behavior of the objective function follows some pattern, which implies that Multi-SAPSO-PRE is effective to solve some practical dynamic optimization problems. The algorithm proposed in this paper may shed light on the optimization of greenhouses.

Future works can take a wide range of aspects into consideration. Evaluation with other dynamic test questions and practical applications would be beneficial. In addition to one of the autoregressive models, polynomial extrapolation, neural networks, Bayesian models, and other predictive methods can be utilized.

Acknowledgements

This work was financially supported by the National 863 High-Tech R&D Program of China under Grant 2013AA103006 and the National Natural Science Foundation of China (Grant No. 61174090), and in part by the U.S. National Science Foundation's BEACON Center for the Study of Evolution in Action, under cooperative agreement DBI-0939454.

References

[1] Y. Jin, J. Branke, "Evolutionary optimization in uncertain environments: A survey," IEEE Trans. Evol. Comput., vol. 9, no. 3, pp. 303-317, 2005.

[2] C. Cruz, J. R. González, D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," Soft Computing, vol. 15, no. 7, pp. 1427-1448, 2011.

[3] J. Kennedy, R. C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, 199, pp. 1942-1948.

[4] Goffe W L, Ferrier G D, Rogers J. Global optimization of statistical functions with simulated annealing. Journal of Econometrics, 1994, 60(1): 65-99.

[5] Luyi Shen, Lihong Xu, Ruihua Wei, Leilei Cao,Multi-swarm Optimization with Chaotic Mapping for Dynamic Optimization Problems, Proceedings of Computational Intelligence and Design (ISCID), 2015, pp. 10.1109/ISCID. 2015.173.IEEE

[6] L. Changhe, S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," Natural Computation, 2008. ICNC'08. Fourth International Conference on. IEEE, vol. 7, pp. 624-628, 2008.

[7] Christopher Chatfield. The Analysis of Time Series: An Introduction. CRC Press, 2004.

[8] Schneider, T. and Neumaier, A. 2001. ARFIT – A Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models, ACM Trans. Math. Soft., Vol. 27(1), March 2001, pp. 58-65.

[9] Li, C., Yang, S., Nguyen, T. T., Yu, E. L., Yao, X., Jin, Y., et al. (2008). Benchmark generator for CEC 2009 competition on dynamic optimization. University of Leicester, University of Birmingham, Nanyang Technological University Technical report.

[10] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," In Congress on Evolutionary Computation CEC99. 1999, pp. 1875–1882.

[11] Brest, J., Korošec, P., Šilc, J., Zamuda, A., Boškovic, ´ B., & Maucec, ˇ M. S. (2013). Differential evolution and differential ant-stigmergy on dynamic optimisation problems. International Journal of Systems Science, 44(4), 663–679

[12] Brest, J., Zamuda, A., Boskovic, B., Maucec, M. S., & Zumer, V. Dynamic optimization using self-adaptive differential evolution. In IEEE congress on evolutionary computation: 2009, pp. 415–422

[13] Liu, L., Wang, D., & Yang, S. (2008). Compound particle swarm optimization in dynamic environments. Applications of evolutionary computing 2008, pp. 616–625. Berlin, Heidelberg: Springer

[14] Lung, R., & Dumitrescu, D. (2007, Sep.). A collaborative model for tracking optima in dynamic environments. In Proceedings of IEEE CEC, September 2007, pp. 564–567.