2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Adaptive Particle Swarm Optimization with Variable Relocation for Dynamic Optimization Problems

Zhi-Hui Zhan, *Member, IEEE*, Jing-Jing Li, and Jun Zhang, *Senior Member, IEEE*

Abstract—**This paper proposes to solve the dynamic optimization problem (DOP) by using an adaptive particle swarm optimization (APSO) algorithm with an variable relocation strategy (VRS). The VRS based APSO algorithm (APSO/VRS) has the following two advantages when solving DOP. Firstly, by using the APSO optimizing framework, the algorithm benefits from the fast optimization speed due to the adaptive parameter control. More importantly, the adaptive parameter and operator in APSO make the algorithm fast respond to the environment changes of DOP. Secondly, VRS was reported in the literature to help dynamic evolutionary algorithm (DEA) to relocate the individual position in promising region when environment changes. Therefore, the modified VRS used in APSO can collect historical information in the stability stage and use such information to guide the particle variable relocation in the change stage. We evaluated both APSO and APSO/VRS on several dynamic benchmark problems and compared with two state-of-the-art DEAs and DEA that also used the VRS. The results show that both APSO and APSO/VRS can obtain very competitive results on these problems, and APSO/VRS outperforms others on most of the test cases.**

I. Introduction

In many real-world optimization problems, dynamics and uncertainties are common characteristics due to the complexity of the systems. For example, the flights may be delayed or even canceled in the aircraft arriving scheduling[1], new sensors may enter the wireless sensor network or old sensors may die [2][3], nodes may increase or connects may change in the multicast routing optimization [4]. This is dynamic optimization problem (DOP) that has caused great attentions and interests in recent years for its significance in practical applications [5]. Different from static optimization problem whose decision variables, objective functions, and problem landscape are unchanged all the time, DOP dooms to be challenging since it requires that the algorithms can not only find the optimal solution in one time, but also have the ability to track the problem changes and find the new optimum when environment changes [6].

Z.-H. Zhan and J. Zhang are with the Department of Computer Science, Sun Yat-Sen University, Guangzhou, 510275, China, with the Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China, with the Engineering Research Center of Supercomputing Engineering Software (Sun Yat-sen University), Ministry of Education, China, and also with the Key Laboratory of Software Technology, Education Department of Guangdong Province, China.

J.-J. Li is with the School of Computer Science, South China Normal University, China.

This work was supported in part by the National High-Technology Research and Development Program (863 Program) of China No.2013AA01A212, in part by the NSFC for Distinguished Young Scholars 61125205, in part by the NSFC No. 61332002 and No.61300044. For additional information regarding this paper please contact Jun Zhang (corresponding author, email: issai@mail.sysu.edu.cn).

Although being challenging, researchers find that evolutionary computation (EC) algorithms are promising in solving DOP. This may be due to that EC algorithms are kinds of stochastic search optimization methods inspired from the biological evolution and swarm intelligence behaviors in nature, making EC algorithms suitable for changing environments [6]. In the EC studies for solving DOP, some researchers proposed to use *population re-initialization approach* to re-initialize the individuals when a change occurred [7]. This is a naïve approach but may be not judicious because restarting the algorithm means throwing away all the previous search information. Therefore, some other researchers proposed to use a kind of memory to store the best individual or some good individuals for each change period, and put these individuals back to the new population in the new search environment if they still had promising performance. Such *memory-based approach* will be promising in cyclic dynamic environment [8]. In some other studies, *multiple population approach* was proposed where several subpopulations were used to track the multiple peaks of the landscape. Such approach may be useful when the global optima are changed (switched) among these multiple peaks. However, they are in large computational cost when compared with single population approach [9]. Besides, *adaptive/self-adaptive approach* was also proposed for solving DOP. For example, the mutation probability can be increased according to the environment changes so as to increase the population diversity for tracking the new optimum. This kind of approach seems to be promising for DOP whose landscape is with very fast but less drastic changes [10].

With various DOP tackling strategies as mentioned above, one important issue is that some recent studies argued that the historical information could be used for helping EC algorithm to locate the new optimum when environment change occurs. Woldesenbet and Yen [11] pointed out that most of the environment changes in practical DOP applications may not be drastic. Therefore, re-initializing the whole population when a change occurs is not efficient for reusing the historical information to fast catch the new search environment. In this sense, a variable relocation strategy (VRS) was proposed for dynamic evolutionary algorithm (DEA) when solving DOP [11]. We confirmed the claims in RVDEA (VRS based DEA) that with the help of VRS, the relocated population was shown to be a better fit to the new environment. Therefore, extensive study of the VRS is conducted in this paper.

Although RVDEA obtained good performance on DOP when compared with some state-of-the-art algorithms, it should be reminded that RVDEA did not use adaptive parameters for better optimizing DOP. As mentioned above, *adaptive/self-adaptive approach* can dynamic adjust the algorithm parameters for match the search requirements of DOP in different evolutionary time. It should be promising if proper adaptive parameter and operator strategies are used for helping solve DOP. Inspired by the VRS idea and the adaptive algorithm control idea for DOP, this paper proposes an efficient DOP approach, based on the powerful adaptive particle swarm optimization (APSO) [12] and the efficient VRS strategy.

PSO is a simple yet efficient EC paradigm that has been fast developed in recent years, mainly due to its simple algorithm concept, easy program implementation, and fast convergence speed to reasonable solution [13],. PSO has been extensively studied in not only the static optimization problem, but also in DOP in recent years [14]. Although promising results can be obtained by PSO in various static/dynamic problems, researchers still find that adaptively control of the algorithm parameters and operators can substantially improve the algorithm performance. Zhan *et al.* [12] proposed the APSO algorithm by using a machine learning based statistical analyze technique to discover the useful information behind the population distribution data and fitness data, so as to design an efficient adaptive parameter and operator control strategy for PSO to improve the performance in different optimization problems and different evolutionary states. This adaptation scheme seems to be much suitable for DOP for that DOP can be regarded as different optimization problems when the landscape changes during the evolutionary progress. Therefore, this paper adopts APSO for solving DOP. Moreover, as VRS has been proven to be efficient in helping EC algorithm to fast track the change environment in DOP, this paper further applied VRS to APSO, resulting in an efficient APSO/VRS algorithm for solving DOP.

The rest of the paper is organized as follows. In Section II, the background including DOP and APSO are briefly described. Section III presents the APSO/VRS algorithm in detail. Experiments and comparisons are conducted in Section IV. Finally, conclusions are drawn in Section V.

II. Background

A. DOP

DOP is a kind of optimization problem whose fitness functions, constraints, or environmental parameters are with possible changes. Specifically, a DOP with maximization objective can be formulated as:

$$\text{Max}\, f(X, e) = f(x_1, x_2, \ldots, x_D, e) \tag{1}$$

where X is the decision vector with D dimensions, and each dimension x_d is with the range $[x_{d\_\min}, x_{d\_\max}]$; f is the objective function to be optimized, and e represents the environmental conditions that are change over the evolutionary time.

B. PSO

PSO is one of the most important swarm intelligence algorithms that was first introduced by Kennedy and Eberhart in 1995. Mimicking the swarm behaviors in birds flocking and fishes schooling, the PSO uses a simple mechanism that lets the particle search for the global optimum in the solution space under the influences of the its own and its companions' experiences. In PSO, each particle i is associated with two vectors, the velocity vector $\boldsymbol{V_i} = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ and the position vector $\boldsymbol{X_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, where $1 \le i \le N$, N is the population size and D is the dimension number of the decision variables. The velocity and the position of each particle are initialized by random vectors as $\boldsymbol{V_i}(0)$ and $\boldsymbol{X_i}(0)$ within the corresponding ranges. In every generation g, the fitness of particle i (denoted as $f_i(g)$) will be evaluated at its current position $\boldsymbol{X_i}(g)$. The best position during the run time is stored to be the personal historical best position $\boldsymbol{Pbest_i} = [p_{i1}, p_{i2}, \ldots, p_{iD}]$. Among all the $\boldsymbol{P_i}$ in the whole swarm, the best one is denoted as the globally best position $\boldsymbol{Gbest} = [g_1, g_2, \ldots, g_D]$. The vectors $\boldsymbol{V_i}$ and $\boldsymbol{X_i}$ are updated by Eqs. (2) and (3) generation by generation through the guidance of $\boldsymbol{Pbest_i}$ and $\boldsymbol{Gbest}$.

$$\begin{aligned} v_{id}(g) = v_{id}(g-1) &+ c_1 r_{1d}(p_{id}(g-1) - x_{id}(g-1)) \\ &+ c_2 r_{2d}(g_d(g-1) - x_{id}(g-1)) \end{aligned} \tag{2}$$

$$x_{id}(g) = x_{id}(g-1) + v_{id}(g) \tag{3}$$

where ω is the inertia weight, c_1 and c_2 are acceleration coefficients, r_{1d} and r_{2d} are two random numbers independently generated within [0, 1] for the d^{th} dimension. Recently, PSO has been studied for performance enhancement by using adaptive strategy [12], orthogonal learning strategy [13], aging mechanism [15], machine learning techniques [16], and for applications as constraint optimization [17], RFID network planning [18], and multi-objective optimization [19].

C. APSO

Given its simple concept and effectiveness, PSO has become a popular optimizer and has been widely applied in practical problems. In traditional PSO, the inertia weight parameter ω is linearly decreasing from 0.9 and 0.4, respectively, while the acceleration coefficients c_1 and c_2 are set to fixed value 2.0. However, the PSO performance is criticized sometimes for the dependences on proper parameter settings. In order to adaptive control the algorithm parameters for release such dependence disadvantages, Zhan *et al.* [12] proposed an APSO to enhance the adaptation of PSO in different problems and different evolutionary states. Herein we briefly describe the APSO algorithm in the following parts.

(1) Evolutionary State Estimation Procedure

In APSO, the algorithm structure and flowchart are similar to those of traditional PSO. However, in each generation, before the update of particle's velocity and position, an evolutionary state estimation (ESE) procedure is first carried out to identify the current state as *exploration*, *exploitation*, *convergence*, or *jumping-out*. The ESE procedure is based on a statistical analyze machine learning technique which can extract useful information from the population distribution data and population fitness data. For self-contain purpose, we briefly describe the ESE procedure as follows: Firstly, calculate the mean distance of each particle i to all the other particles according to the Euclidian distance by:

$$d_i = \frac{1}{N-1}\sum_{j=1, j\neq i}^{N}\sqrt{\sum_{k=1}^{D}(x_i^k - x_j^k)^2} \tag{4}$$

Then calculate the '*evolutionary factor*' *ef* as defined by:

$$ef = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \in [0, 1] \tag{5}$$

where d_g is the d_i of the globally best particle, while $d_{\max}$ and $d_{\min}$ are the corresponding d_i of the particles with the maximal and minimal d_i values among all the particles.

At last, classify the current evolutionary state as S_1 (*exploration*), S_2 (*exploitation*), S_3 (*convergence*), or S_4 (*jumping-out*) based on a fuzzy classification strategy. The corresponding membership functions of the four fuzzy sets are illustrated in Fig. 1. According to the membership functions, the classification works as the following three rules:

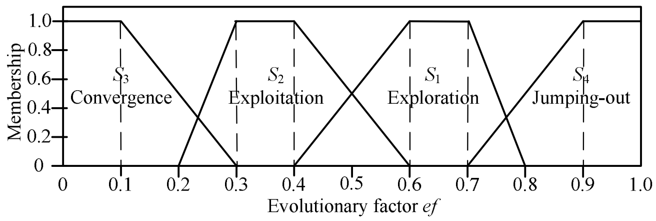

Figure 1. Fuzzy membership functions for four fuzzy evolutionary states.

Rule 1: *The unique rule*. If the *ef* value locates in a position which is unique to a set, then classify the state as the corresponding state. For example, *ef*=0.1 is unique to the S_3 and *ef*=0.65 is unique to the S_1. However, if the *ef* value belongs to two sets, then we first classify the state first based on Rule 2, and then Rule 3.

Rule 2: *The stability rule*. This rule classifies the state to the same state as the last generation. For example, when *ef*=0.45, the state can be S_1 or S_2, in this condition, if the last state is S_1, the current state is set to S_1, if the last state is S_2, the current state is set to S_2.

Rule 3: *The neighborhood rule*. When Rule 2 can not classify the state, for example, when *ef*=0.45, but the last state is neither S_1 nor S_2. In this condition, the states are classified based on the ideal transfer sequence as $S_1 \Rightarrow S_2 \Rightarrow S_3 \Rightarrow S_4 \Rightarrow S_1 \Rightarrow$... So, as *ef*=0.45 indicates the current state can be S_1 or S_2, if the last state is S_3, the current state is set to S_2 because is the neighbor of S_3, if the last state is S_4, the current state is set to S_1.

(2) Parameters Adaptation and Elitist Learning Strategy

After the ESE procedure, the APSO parameter ω, c_1, and c_2 are adaptively controlled. According to the paper [12], the value of the inertia weight ω is initialized to 0.9 and adaptively set through identifying the state of evolution by the sigmoid mapping as:

$$\omega(ef) = \frac{1}{1+1.5e^{-2.6\cdot ef}} \in [0.4,\ 0.9],\ \forall ef \in [0,1] \tag{6}$$

For the acceleration coefficients c_1 and c_2, they are both initialized as 2.0 and adaptively controlled according to different evolutionary states by the strategies described as [12]: "*increasing c_1 and decreasing c_2 in an exploration state, increasing c_1 slightly and decreasing c_2 slightly in an exploitation state, increasing c_1 slightly and increasing c_2 slightly in a convergence state, decreasing c_1 and increasing c_2 in a jumping-out state.*" Herein, the increment and decrement is a random generated value in range [0.05, 0.1]. However, if the "slightly" is used, the generated value is multiplied 0.5 before adds to or subtract from the last c_1 and c_2 values. It should be noted that the values of c_1 and c_2 are clamped in range [1.5, 2.5] and their sum is clamped in range [3.0, 4.0], refer to [12] for more details.

Besides the above parameter adaptation, APSO has an adaptive elitist learning strategy (ELS) which is only carried out in the *jumping-out* state for the globally best particle to jump out possible local optima. The ELS is to randomly choose one dimension of globally best particle's historical best position, and then applied to it through a Gaussian perturbation. If the new position is with better fitness value than the current *gBest*, it will be accepted to replace the current *gBest*. Otherwise, the new position can be used to replace the particle with the worst fitness in the swarm. For more details of the ELS, please refer to [12].

III. Proposed APSO/VRS Algorithm

In this section, the proposed APSO/VRS algorithm for solving DOP is described. Firstly, the VRS operations are presented, including the *historical information collection in the stability stage* and the *particle variables relocation in the change stage*. Then, the whole complete APSO/VRS algorithm flowchart is presented.

A. VRS

The VRS was proposed by Woldesenbet and Yen [11] in DEA to make the use of historical search information to guide the individuals to relocate their position when a change occurs. Here in this paper, when applying VRS to APSO, some modifications should be made to match the characteristics of PSO and APSO. In the following context, the DOP is assumed to be a maximization problem as (1). The VRS used in APSO is described as "*historical information collection in the stability stage*" and "*particle variables relocation in the change stage*".

(1) Historical Information Collection in the Stability Stage

In every generation during the APSO process when the environment is static, VRS collects the progresses of each particle's position and fitness during the run. The "*dimension progress*" for the d^{th} dimension of particle i is defined as:

$$\Delta x_{id}(g) = x_{id}(g) - x_{id}(g-1) \tag{7}$$

where g=1, 2, 3,... is the current generation index. It is interesting to notice that the $\Delta x_{id}(g)$ is actually the $v_{id}(g)$ in (3).

Besides the "*dimension progress*", the "*fitness progress*" of particle i is defined as:

$$\Delta f_i(g) = f_i(g) - f_i(g-1) \tag{8}$$

Based on the $\Delta x_{id}(g)$ and $\Delta f_i(g)$, the "*average dimension progress*" $\overline{\Delta x_{id}}(g)$ and "*average fitness progress*" $\overline{\Delta f_i}(g)$ of the g^{th} generation along the evolutionary process are defined as (9) and (10):

$$\overline{\Delta x_{id}}(g) = \frac{\Delta x_{id}(g) + \lambda \overline{\Delta x_{id}}(g-1)}{\lambda g + 1} \tag{9}$$

$$\overline{\Delta f_i}(g) = \frac{\Delta f_i(g) + \lambda \overline{\Delta f_i}(g-1)}{\lambda g + 1} \tag{10}$$

where $\overline{\Delta x_{id}}(0)$ and $\overline{\Delta f_i}(0)$ are initialized as 0. λ is a parameter to control the influences of historical progress. When λ=1, it means the "average" is actual the arithmetic average of the progresses from the beginning to now. When λ=0, it means the "average" is just the current progress, without the consideration of progresses in previous generations. Herein we adopt λ=0.5 as recommended in [11] to indicate the historical information is considered but the influences reduce as the time goes by.

It should be reminded that the operations (7)-(10) are carried out in every generation between any two changes. It is also interesting to remind that the result of (7) is actually the velocity of the current generation.

(2) Particle Variables Relocation in the Change Stage

When a change occurs, the above information collected during the stability stage is used to guide particle variables relocation. The operations are described as the following 7 steps.

Step 1: Calculate the "*average position progress*" of particle *i* by considering its all dimensions. That is, using the information in (9) to obtain the information in (11) as:

$$\overline{\Delta X_i} = \sqrt{\sum_{d=1}^{D} \left(\overline{\Delta x_{id}}\right)^2} \tag{11}$$

As the step is carried out only in the environment change time, the generation index g is not used in the following equations. Therefore the $\overline{\Delta x_{id}}$ in (11) is the $\overline{\Delta x_{id}}$ of the last generation just before the change occurs. Such situations are similar in the following equations.

Step 2: Calculate the "*average sensitivity*" as the ratio of "*average fitness progress*" in the objective space to the "*average position progress*" in the decision space. That is, using the information in (10) and (11) to obtain the information in (12), as:

$$\overline{S^{X_i}} = \frac{\overline{\Delta f_i}}{\overline{\Delta X_i}} \tag{12}$$

Step 3: Calculate the "*average sensitivity*" of the d^{th} dimension as (13) which is obtained from the information of (12), (9), and (11), as:

$$\overline{s^{x_{id}}} = \overline{S^{X_i}} \cdot \frac{\overline{\Delta x_{id}}}{\overline{\Delta X_i}} \tag{13}$$

Step 4: Obtain the actual "*fitness difference*" of particle *i* in the change as:

$$\Delta f_i = f_i^{e\_new} - f_i^{e\_old} \tag{14}$$

where $f_i^{e\_new}$ and $f_i^{e\_old}$ are the particle *i*'s fitness values after and before change respectively.

Step 5: Obtain the relocation radius as:

$$\Delta R_i = \begin{cases} -\dfrac{\Delta f_i}{\overline{S^{X_i}}}, & , \quad f_i^{e\_new} \le f_i^{e\_old} \\ \min\left\{ \dfrac{f_{best}^{e\_new} - f_i^{e\_old}}{\overline{S^{X_i}}}, \dfrac{\Delta f_i}{\overline{S^{X_i}}} \right\}, & f_i^{e\_new} > f_i^{e\_old} \end{cases} \tag{15}$$

where $f_{best}^{e\_new}$ is the best fitness values in the new environment.

Step 6: Obtain the relocation radius for each dimension as:

$$\Delta r_{id} = \frac{\Delta R_i \cdot \overline{s^{x_{id}}}}{\overline{S^{X_i}}} = \Delta R_i \cdot \frac{\overline{x_{id}}}{\overline{X_i}} \tag{16}$$

Step 7: Relocate the position of particle *i* as:

$$x_{id}^{new} = x_{id}^{old} + p \cdot \Delta r_{id} \tag{17}$$

where *p* is a random number between [0, 1].

For more details about the constraints of Δr_{id} and x_{id}^{new} please refer to [11].

B. The Complete Flowchart

The complete flowchart of APSO/VRS is shown in Fig. 2. It should be noted that APSO/VRS uses an archive with size 3 to store some best particles and their fitness values. In every generation, if the fitness values change while the positions do not change, the environment is regarded as change.

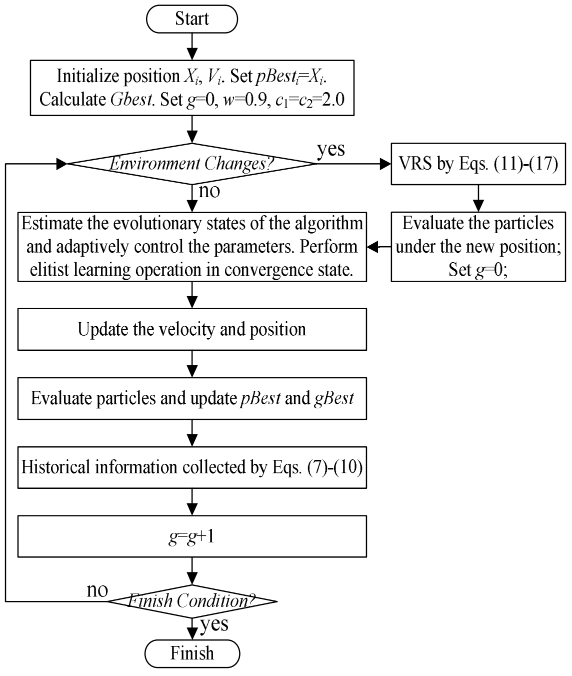
Start
Initialize position X_i, V_i. Set $pBest_i$=X_i. Calculate $Gbest$. Set g=0, w=0.9, c_1=c_2=2.0
Environment Changes?
yes
no
VRS by Eqs. (11)-(17)
Evaluate the particles under the new position; Set g=0;
Estimate the evolutionary states of the algorithm and adaptively control the parameters. Perform elitist learning operation in convergence state.
Update the velocity and position
Evaluate particles and update $pBest$ and $gBest$
Historical information collected by Eqs. (7)-(10)
g=g+1
no
Finish Condition?
yes
Finish

Figure 2. The flowchart of APSO/VRS.

IV. Benchmark Tests and Comparisons

A. Experimental Settings

In this paper, 5 typical DOPs with different characteristics are used to test the performance of APSO/VRS. The 5 DOPs are the MP1 and DF2-DF5 as detailed in [11]. In order to investigate the advantages of APSO, we will compare APSO/VRS with RVDEA proposed in [11]. Moreover, APSO/VRS is also compared with APSO without using the VRS, to demonstrate the influences of VRS on APSO/VRS. It should be noted that the results of RVDEA are directly adopted from [11]. They were actually obtained by RVDEA/Mem and RVDEA/Cluster which were both enhanced RVDEA. In order to make fair comparisons, we use the same performance metric, problem parameters, and computational burden as in [11] to evaluate APSO and APSO/VRS.

Firstly, the performance metric is the offline error variation, which is the average of the error between the true optimal fitness and the best fitness at each function evaluation (FE):

$$\overline{e_{offline}} = \frac{1}{FEs}\sum_{fe=1}^{FEs}\left|f_{true} - f_{best}(fe)\right| \qquad (18)$$

Secondly, all the 5 DOP functions are with the same problem parameters such as the search range, default peaks number, dimensions, change frequency, and change severity, as set in [11]. One thing should be noted is that the default change frequency is set to 5000 FEs [11].

Thirdly, APSO and APSO/VRS use population with size 20, the maximal FEs is 500000, the same in [11]. The APSO parameters are the same as proposed in [12]. Moreover, the results are the average of 50 independent runs as in [11].

B. Experimental Results

For the MP1 problem, the peak number is set to be 1, 10, 20, 30, 40, 50, 100, and 200, respectively. Table I presents the results of different DOP algorithms when solving MP1 with different peaks number. The best results are in **boldface**.

TABLE I. MEAN OFFLINE ERROR OF DOP ALGORITHMS ON MP1 WITH DIFFERENT PEAKS NUMBER

Peaks no.	RI25/ Mem	HM/ Mem	RVDEA/ Mem	RVDEA/ Cluster	APSO	APSO/ VRS
1	9.28	11.98	1.23	1.02	17.567	**0.777**
10	14.63	15.62	4.88	3.54	15.395	**0.798**
20	13.87	16.02	5.68	3.87	13.460	**0.777**
30	12.89	14.24	5.86	3.92	13.447	**0.747**
40	12.41	13.93	5.65	3.49	13.687	**0.759**
50	12.74	13.97	5.21	3.78	12.223	**0.789**
100	11.20	13.81	4.98	3.37	11.847	**0.736**
200	10.82	14.06	4.92	3.54	10.933	**0.736**

Results of RI25/Mem, HM/Mem, RVDEA/Mem, and RVDEA/Cluster are from [11].

TABLE II. MEAN OFFLINE ERROR OF DOP ALGORITHMS ON MP1 WITH 10 PEAKS AND DIFFERENT CHANGE FREQUENCIES

Cha. Freq. (FEs)	RI25/ Mem	HM/ Mem	RVDEA/ Mem	RVDEA/ Cluster	APSO	APSO/ VRS
200	21.34	20.12	15.62	15.82	35.084	**13.394**
500	19.13	19.15	8.59	8.89	33.707	**7.031**
1000	18.78	18.84	6.51	7.21	29.615	**4.087**
2500	16.70	17.10	4.93	5.35	21.722	**1.530**
5000	14.63	15.62	4.01	4.88	14.957	**0.759**
10000	13.79	14.23	3.62	4.12	9.590	**0.417**

TABLE III. MEAN OFFLINE ERROR OF DOP ALGORITHMS ON DF2 WITH DIFFERENT PEAKS NUMBER

Peaks no.	RI25/ Mem	HM/ Mem	RVDEA/ Mem	RVDEA/ Cluster	APSO	APSO/ VRS
1	9.05	10.16	1.79	0.302	2.25	**0.103**
5	11.91	12.42	4.2	2.653	1.59	**0.105**
10	13.22	14.37	6.36	3.871	1.72	**0.098**
50	14.11	16.38	7.54	3.322	1.27	**0.082**
100	17.52	17.66	8.06	3.713	1.14	**0.083**
200	20.64	21.2	11.59	3.755	0.97	**0.073**

TABLE IV. MEAN OFFLINE ERROR OF DOP ALGORITHMS ON DF2 WITH 10 PEAKS AND DIFFERENT CHANGE FREQUENCIES

Cha. Freq. (FEs)	RI25/ Mem	HM/ Mem	RVDEA/ Mem	RVDEA/ Cluster	APSO	APSO/ VRS
200	21.15	20.76	14.41	10.11	9.49	**0.871**
500	17.86	19.24	8.83	7.55	7.03	**0.414**
1000	15.99	16.28	6.98	4.41	5.15	**0.230**
2500	14.6	15.56	6.44	4.12	2.44	**0.134**
5000	13.22	14.37	6.36	3.87	1.50	**0.087**
10000	11.53	12.43	5.95	3.34	0.99	**0.068**

TABLE V. MEAN OFFLINE ERROR OF DOP ALGORITHMS ON DF3-DF5 WITH DIFFERENT PEAKS NUMBER

Problem	Peaks no.	RI25/ Mem	HM/ Mem	RVDEA/ Mem	RVDEA/ Cluster	APSO	APSO/ VRS
Linear DF3	1	7.8	8.32	1.517	**0.081**	1.072	0.541
	5	7.97	8.41	1.892	1.122	1.044	**0.575**
	10	8.31	8.54	2.082	1.609	1.136	**0.673**
	50	8.64	8.69	2.367	1.756	1.002	**0.659**
	100	8.73	8.82	2.688	2.186	1.143	**0.668**
	200	8.88	9.02	2.85	2.377	1.015	**0.636**
Random DF4	1	9.21	9.19	1.268	**0.106**	0.514	0.351
	5	9.44	9.53	1.764	1.446	1.836	**1.195**
	10	9.64	9.71	2.303	1.791	**1.255**	1.300
	50	9.67	10.03	2.675	1.862	0.999	**0.712**
	100	10.32	10.55	2.904	1.985	0.715	**0.615**
	200	10.56	10.78	3.023	2.149	0.551	**0.499**
Circular DF5	1	9.86	10.08	1.687	**0.158**	0.611	0.397
	5	10.1	10.39	2.022	0.967	0.603	**0.478**
	10	10.61	10.83	2.445	1.162	0.594	**0.494**
	50	10.7	10.9	2.664	1.368	0.581	**0.467**
	100	10.92	10.95	2.864	1.743	0.597	**0.510**
	200	11.1	11.18	3.191	2.04	0.590	**0.490**

TABLE VI. MEAN OFFLINE ERROR OF DOP ALGORITHMS ON DF3-DF5 WITH 10 PEAKS AND DIFFERENT CHANGE FREQUENCIES

Problem	Cha. Freq. (FEs).	RI25/ Mem	HM/ Mem	RVDEA/ Mem	RVDEA/ Cluster	APSO	APSO/ VRS
Linear DF3	200	8.7	8.83	2.334	0.881	9.073	**0.706**
	500	8.42	8.69	2.125	0.755	2.222	**0.599**
	1000	8.31	8.54	2.083	**0.609**	1.084	0.669
	2500	8.17	8.38	1.781	**0.483**	0.732	0.574
	5000	8.08	8.22	1.622	**0.299**	0.623	0.565
	10000	7.9	8.11	1.413	**0.177**	0.692	0.584
Random DF4	200	9.93	9.88	2.752	**1.026**	6.205	6.874
	500	9.76	9.78	2.611	**0.982**	2.352	2.926
	1000	9.64	9.71	2.303	**0.891**	1.295	1.023
	2500	9.42	9.59	2.142	0.769	0.826	**0.709**
	5000	9.28	9.46	1.897	0.536	0.519	**0.479**
	10000	9.17	9.33	1.662	**0.247**	0.376	0.392
Circular DF5	200	10.9	11.05	2.989	1.583	0.786	**0.572**
	500	10.75	10.96	2.788	1.457	0.648	**0.466**
	1000	10.61	10.83	2.445	1.162	0.646	**0.425**
	2500	10.47	10.72	2.121	0.783	0.556	**0.449**
	5000	10.29	10.53	1.965	0.607	0.645	**0.539**
	10000	10.22	10.41	1.792	**0.340**	0.433	0.493

Results of RI25/Mem, HM/Mem, RVDEA/Mem, and RVDEA/Cluster are from [11].

The results in Table I show that APSO/VRS obtained the best results on MP1 with different peaks number. The related good results of RVDEAs also indicate the advantages of VRS. However, APSO/VRS is much better than RVDEAs, demonstrating the advantages of combining both APSO and VRS. The experiments on MP1 with 10 peaks under different change frequencies were also conducted and the results are presented in Table II. The table shows that the offline error gets to be better and better as the change becomes less frequent. This may be due to that the algorithm has enough time to converge to the global optimum between any two changes. The notable observation is that our proposed APSO/VRS has the best performance among all the algorithms under different environment change frequencies, no matter fast or slow.

The results on DF2 with different peaks number and the results on DF2 with 10 peaks under different change frequencies are presented in Tables III&IV, respectively. The results in Table III show that with the peak number increasing, the performance of traditional DEAs and RVDEAs becomes poorer and poorer. However, things are different in APSO and APSO/VRS. The performance is less influenced by the peaks number, and is even observed to be better with a higher number of peaks. This is an interesting observation in APSOs and such advantages make the APSOs algorithm suitable for complex DOPs. Moreover, the results in Table IV show that both APSO and APSO/VRS outperform both the RVDEA/Mem and RVDEA/Cluster under different change frequencies for DF2, expect that RVDEA/Cluster is slightly better than APSO when the frequency between two changes is 1000 FEs and 2000 FEs. APSO/RVS obtained the best results on all the test cases with its results greatly better than those of RVDEAs.

DF3, DF4, and DF5 are three DOPs with the same based functions but with different dynamic characteristics, as linearly, random, and circularly varying moving, respectively. In Tables V&VI, the results on DF3-DF5 with different peaks number and problems with 10 peaks under different change frequencies are presented respectively. It shows that RVDEA/Cluster only outperforms APSO/VRS on the problems with 1 peak, while APSO/VRS archives much better results than RVDEAs and other algorithms when the peaks number increases. Moreover, APSO/VRS can obtain competitive results on the problems under different change frequencies. The performance of APSO/VRS is much better when dealing with DF5.

V. Conclusion

In this paper, we have applied the VRS to APSO to develop a promising algorithm for solving DOPs. The APSO algorithm itself is suitable for tracking the changing environment of DOP because it can online adaptively control the parameters and operator to accelerate the optimization speed and increase the population diversity according to different search requirements. Moreover, VRS can collect the historical information in the stability stage and use the information in the change stage to guide the particles to relocate in promising position for fast tacking the new global optimal solution. Experiments have been conducted on 5 typical DOP benchmarks with different characteristics, to demonstrate not only the APSO promising performance but also the APSO/VRS outstanding performance.

References

[1] Z. H. Zhan, J. Zhang, Y. Li, O. Liu, S. K. Kwok, W. H. Ip, and O. Kaynak, "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 399-412, 2010.

[2] Z. H. Zhan, J. Zhang, K. J. Du, and J. Xiao, "Extended binary particle swarm optimization approach for disjoint set covers problem in wireless sensor networks," in *Proc. Conf. Technologies and Applications of Artificial Intelligence*, Nov. 2012, pp. 327-331.

[3] Z. H. Zhan, J. Zhang, and Zhun Fan, "Solving the optimal coverage problem in wireless sensor networks using evolutionary computation algorithms," in *Proc. of the 8th Int. Conf. Simulated Evolution and Learning*, 2010, pp. 166-176.

[4] M. Shen, Z. H. Zhan, W. N. Chen, Y. J. Gong, J. Zhang, and Y. Li, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks," *IEEE Trans. Ind. Electron*. In press, 2014.

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

[6] T. T. Nguyen, S. X. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1-24, 2012.

[7] K. Krishnakumar, "Micro-genetic algorithms for stationary and nonstationary function optimization," in *Proc. SPIE Conf. Intell. Control and Adaptive Syst.*, 1989, pp. 289–296.

[8] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1875–1882.

[9] T. Blackwell and J. Branke, "Multi-swarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Oct. 2006.

[10] T. Nanayakkara, K. Watanabe, and K. Izumi, "Evolving in dynamic environments through adaptive chaotic mutation," in *Proc. Int. Symp. Artif. Life Robot.*, 1999, pp. 520–523.

[11] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Trans. Evol. comput.*, vol. 13, no. 3, pp. 500-513, Jun. 2009.

[12] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, and Cybern. B.*, vol. 39, no. 6, pp. 1362-1381, Dec. 2009.

[13] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832-847, Dec. 2011.

[14] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959-974, Dec. 2010.

[15] W.Chen, J. Zhang, Y.Lin, N.Chen, Z. Zhan, H.Chung, Y. Li, and Y. Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241-258, 2013.

[16] J. Zhang, Z. H. Zhan, Y. Lin, N. Chen, Y. J. Gong, J. H. Zhong, H. Chung, Y. Li, and Y. H. Shi, "Evolutionary computation meets machine learning: A survey," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 68-75, Nov. 2011.

[17] Y. J. Gong, J. Zhang, H. Chung, W. N. Chen, Z. H. Zhan, Y. Li, and Y. Shi, "An efficient resource allocation scheme using particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 6, pp. 801-816, Dec. 2012.

[18] Y. J. Gong, M. Shen, J. Zhang, O. Kaynak, W. N. Chen, and Z. H. Zhan, "Optimizing RFID network planning by using a particle swarm optimization algorithm with redundant reader elimination," *IEEE Trans. Ind. Informa.*, vol. 8, no. 4, pp. 900-912, Nov. 2012.

[19] Z. H. Zhan, J. Li, J. Cao, J. Zhang, H. Chung, and Y. H. Shi, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445-463, April. 2013.