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

Niching-based Self-adaptive Ensemble DE with MMTS for solving Dynamic Optimization Problems

Sheldon Hui and Ponnuthurai Nagaratnam Suganthan
School of Electrical and Electronics Engineering
Nanyang Technological University
Singapore, 639798
{Shel0003,epnsugan}@ntu.edu.sg

***Abstract*— Dynamic and non-stationary problems require optimization algorithms search for the best solutions in a time-varying fitness environment. Various methods and strategies such as niching, clustering and sub-population approaches have been implemented with Differential Evolution (DE) to handle such problems. With the help of crowding niching to maintain general population diversity, this paper attempts to extend the Self-adaptive Ensemble DE with modified multi-trajectory search attempt to solve CEC2014 dynamic optimization competition benchmark problems.**

Keywords - Dynamic Optimization Problems (DOPs), Self-adaptive Ensemble Differential Evolution (SaDE), modified multi-trajectory search (MMTS), crowding

I. INTRODUCTION

Real-world optimization applications are time-varying in nature, and benchmark problems simulate this by generating changes to the fitness landscape over a unit time defined by number of functional evaluations (FEs). Such dynamic optimization problems (DOPs) require optimizers to adapt to the changes continuously because of the changing optima. With stationary problems, the algorithmic behavior of optimizers tends towards abandoning exploration for exploitation over iterations. This cannot hold true for dynamic problems because the locations of the true optima may shift dynamically with no predictable direction. Also the dimensions of the objective function are subject to change [1]. As such, modifications must be made to the state-of-the-art techniques to handle such problems [1, 6].

Advances have been made in Evolutionary Algorithms (EAs) class of population-based optimizers to solve DOPs in both Particle Swarm Optimizers (PSOs) and Differential Evolution (DE) [2]. PSO imitates the behavior of birds or insect swarms while DE simulates evolution on a population of real-valued vectors. Both techniques systematically scour the fitness landscape for feasible regions where the true optima are, but inevitably loose their population diversity when converging onto a solution [4, 5]. The loss of population diversity, however, is a liability when solving dynamic problems because it results in stagnation when dynamic changes occur [6]. The obvious solution is to completely re-randomize the population, but this would in turn means loss of any useful information about the changing environment, particularly if the changes are small or recurrent [1, 6].

DE mutation and crossover strategies are known to be both diverse and specialized for exclusive fitness optimization situations. Since there can be no one-size-fits-all strategy or parameter in the case of dynamic optimization, the self-adaptive ensemble strategy and parameters DE (SaEPSDE) seems to be a prospective candidate for handling dynamic optimization problems. Complementing the SaEPSDE is the modified multi-trajectory search (MMTS) introduced by Zhang and Sanderson [3] which helps to enhance multiple diverse solutions at different evolution stages. Even so, global population diversity in DE would ultimately diminish over iterations, and this is where the niching mechanism plays its part in maintaining global population diversity in anticipation of dynamic changes. Past best solutions are stored in an archive after each change occurred and clearing is applied to ensure population diversity in the archive of past best solutions.

This paper is structured as follows: Section II describes past research in dynamic optimization. Section III introduces the basic DE algorithm. Section IV explains our proposed algorithm. Section V provides details of our experimental results from the CEC2014 Generalized Dynamic Benchmark Generator (GDBG) benchmark problems [7]. Finally, Section VI concludes the paper.

II. RELATED WORKS

This section summarizes past developments in dynamic optimization using EAs and various techniques developed for handling of dynamic changes.

A. Dynamic Optimization Problems

The generalized definition of the dynamic optimization problem is as follows [6, 7]:

$$F := f(\mathbf{x}, t) \tag{1}$$

where $\mathbf{x} = [x_1, x_2, \ldots x_D]$ is the D-dimensional solution vector of parameters, and t is time, although the unit of t may indeed be different from actual real time for practical purposes. The most typical measure of t in dynamic benchmark is the number of functional evaluations of the problem's fitness, although on some occasions iterations of the optimizing algorithm's runtime over 1 cycle may be used. Regardless of the definitions of time in the case of dynamic optimization, the

state of the fitness landscape in a dynamic problem is triggered to change over fixed samples of the time that an algorithm is allowed to run.

DOP benchmark generators for simulating dynamic changes in the fitness landscape had been designed for DOP optimizers. The 1999 Congress on Evolutionary Computation (CEC'99) introduced two dynamic problems simulators: the highly popular Moving Peaks Benchmark (MPB) by Branke [8], and the DF1 bench-mark generator by Morrison and De Jong [9]. In 2009, the Generalized Dynamic Benchmark Generator (GDBG) by Li and Yang was adopted as the benchmark for the CEC'09 Competition on Dynamic Optimization [7], and in CEC'14, the GDBG is expanded to include more change instances to better emulate real-world dynamic problems.

B. Change handling of EAs

a) Change detection

Handling changes in a dynamic environment require algorithms to first detect any changes to the problem environment. The easiest and most practical way to do this is to re-evaluate one or more members of the population over iterations before evolution is conducted [10]. The general assumption here is that only one instance of change would occur within an iteration of the population. Another assumption made in this manner of detection is that the general population is population diversely distributed throughout the fitness landscape so that partial changes to the environment could also be detected [1, 6]. Since EAs loose population diversity over iterations, many approaches have been employed to maintain at least some degree of population diversity over time. One method of doing so is to designate fixed locations in the fitness landscape to be re-evaluated for change over iteration [6]. Other techniques simply attempt to maintain the population diversity throughout the execution of evolution. [8, 10 - 16]

b) Complete reinitialization: Early dynamic optimization techniques based on Genetic Algorithms (GAs) simply perform complete random re-initialization on the population when a dynamic change occurs [6]. This is highly inefficient due to total loss of information about the environment prior to change [1].

c) Memory-based Approaches: Since dynamic changes may be slight or recurrent in nature, the location of the new global optima after change may end up close in proximity to past optimal solutions. It is therefore pertinent to store the locations of past global and local optima in an archive. This information can then be used as a reinitialization pool for the new population [10, 11], either directly or with some form of modifications [12]. The Clustering PSO (CPSO) by Yang and Li (2010) stores past solutions in a cradle swarm from which new swarm clusters are generated [10]. In the Self-adaptive DE (jDE), past solutions are re-diversified with additive noise before being applied for reinitialization [12].

d) Single-population diversity maintenance: Since loss of population diversity is the key challenge in EAs for handling dynamic changes, many diversity maintenance approaches have been developed to [11, 19] ensure that complete loss of population diversity does not occur. The Composite Particle PSO (PSO-CP) by Liu, Yang and Wang maintain overall population diversity by modelling every 3 particles in their population after the nature of composite element particles [19]. The dynamic Evolutionary Programming (dynEP) by E. L. Yu and P. N. Suganthan (2009) partially or completely rediversifies a converged population by observing the fitness standard deviation in the population [11].

e) Multi-population approaches: This is generally the most popular technique for dynamic optimization because the goal of overall population diversity is in line with the need for local population convergence. Multi-population approach and hybrid-variants are often complemented with other diversity maintenance techniques involving explorative mutation strategies and parameters. Examples of these are the Multi-swarms with exclusion and anti-Convergence of Blackwell and Branke [13], the Species-based PSO (SPSO) by Parrott and Li (2006) [14]. Similarly for DE, Mendes and Mohais introduced the Dynamic DE (DynDE) with the same diversity maintenance structures of Multi-swarms [15]. This was further enhanced by Plessis and Engelbrecht by introducing favored populations and migrating individuals [16]. Clustering and niching techniques can also be considered as forms of multi-population techniques [18].

III. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is introduced by Storn and Price in 1995 as an alternative EA utilizing weighted difference between two or more random individuals for mutation and parent-mutant crossover. Selection is conducted between parent and offspring populations of real-valued vectors [4, 5]. The conventional DE parameters are the population size NP, scaling factor F and crossover probability CR.

In classical DE, a population of size NP is first randomly initialized in a continuous search space of dimension D, $\boldsymbol{R}$ bounded upper and lower by, x_j^U and x_j^L respectively. The ith individual, $\mathbf{x}_i = [x_{1,i}, x_{2,i}, \ldots x_{D,i}] \in \boldsymbol{X}$, i=1,2,…,N, is mutated and crossed-over and updated using parent-offspring selection. The fitness of the population is computed with objective function f: $D \subseteq \boldsymbol{R}^n \rightarrow \boldsymbol{R}$, and $D \neq \Phi$, and fitness is defined either as a minimization or a maximization problem. The process of mutation and crossover is described as follows.

A. Mutation

In the classic DE, the parent (target) population is perturbed into the mutant vectors population, **V,** of size NP. Various mutation strategies have been developed with their individual advantages and disadvantages. Some of the most popular strategies are as follows:

DE/best/1

$$\mathbf{v}_i^G = \mathbf{x}_{best}^G + F(\mathbf{x}_{rand1,i}^G - \mathbf{x}_{rand2,i}^G) \quad (2)$$

DE/rand/1

$$\mathbf{v_i}^{G} = \mathbf{x}_{\text{rand1,i}}{}^{G} + F(\mathbf{x}_{\text{rand2,i}}{}^{G} - \mathbf{x}_{\text{rand3,i}}{}^{G}) \quad (3)$$

where $\mathbf{x}_{best}$ is the current fittest individual and $\mathbf{x}_{\text{rand1}}$ to $\mathbf{x}_{\text{rand5}}$ are randomly selected individuals with no replacement from the population. The generation number is denoted by G, while the scaling factors F and K determine the step-size in the search direction dictated by the difference vectors [4].

B. Crossover

Crossover is performed between the mutant vectors and their corresponding target vectors. The resultant vector U is known as the trial vector. Two methods of crossover have been proposed, and they are as follows:

Binomial crossover

$$U_{i,j} = \begin{cases} \mathrm{v}_{i,j} \; if \; (rand_j[0,1) \le CR) \text{ or } (j = j_{rand}) \\ x_{i,j} \qquad\qquad otherwise \end{cases} \quad (4)$$

where $U_{\text{i,j}}$ refers to the jth dimension in the ith trial vector. CR is the probability of crossover from the target vector. For Binomial crossover, each dimension has a random chance of crossover with the parent vector, with a minimum of at least 1 random dimension (j=j_{rand}) being inherited from the trial vector.

Exponential crossover

$$U_{i,j} = \begin{cases} v_{i,j}\,, \text{ for } j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ x_{i,j}\,, \text{for all other } j \in [1, D] \end{cases} \quad (5)$$

Exponential crossover, on the other hand randomly selects the nth dimension as starting point to begin crossover (j = $\langle n \rangle_D$) from the trial vector. Subsequent dimensions (j = $\langle n+1 \rangle_D, \ldots \langle n+L-1 \rangle_D$) after n are subject to inherit from trial vector with probability CR. L indicates the total number dimensions inherited from trial vector.

C. Selection

Finally, the trial vector population is pit against the original parent population. The fitness of each member of the trial population is compared with its corresponding target vector, and the fitter member survives into the next generation. Over generations, this evolution process propels the population towards the best regions in the problem space where optimal solutions may be discovered.

$$x_i^{G+1} = \begin{cases} u_i^G, if \; f(u_i^G) \le f(x_i^G) \\ x_i^G, otherwise \end{cases} \quad (6)$$

IV. SELF-ADAPTIVE ENSEMBLE DE WITH MMTS (SAEPSDE-MMTS)

SAEPSDE-MMTS complements the self-adaptation mechanism of SaDE-MMTS [20] with the ensemble methods of Ensemble parameters and strategies DE (EPSDE) by Mallipeddi et. al. [17] so as to allow greater generalization in optimizing single-objective problems. The modified multi-trajectory search was first introduced by Tseng and Chen [23], and then hybridized with SaDE to solve large-scale single-objective numerical problems [20]. SaDE automatically adjusts the primary DE parameters of F and CR values based on the computed probability of successive parent-offspring replacements over iterations. MMTS, on the other hand, perform multi-step sized search along each dimension of the parameter space [23, 24]. Taken together, the SaDE-MMTS was demonstrated to be highly effective for solving large-scale numerical optimization problems.

As mentioned in Section III, different mutation strategies in DE have their distinct advantages and disadvantages. It is therefore imperative to select from a few effective strategies adaptively, particularly when the problem changes dynamically [17]. Since SaDE consisted of only 1 mutation strategy adapted from JADE [26], it was decided that even greater generalization may be achieved by applying ensemble strategies and parameter settings. The SaEPSDE-MMTS has been demonstrated to have achieved statistically significant improvement over SaDE-MMTS within iterations [26]. This paper attempts to extend the algorithm further by applying niching concepts for diversity maintenance necessary in dynamic optimization.

A. SAEPSDE-MMTS

Mallipeddi et. al. (2011) introduced the Ensemble parameters and strategy DE (EPSDE) with the ensemble of strategies and parameters as defined below:

1) Scaling Factor $F \in$ [0.3, 0.7, 0.9]
2) Crossover Ratio $CR \in$ [0.1, 0.5, 0.9]
3) Crossover Methods:
   a) Binomial crossover
   b) Exponential crossover
4) Mutation Strategies:
   a) Current-to-$p$best/2

$$\mathbf{v_i}^{G} = \mathbf{x_i}^{G} + (1 - F_i)(\mathbf{x}_{\text{pbest}}{}^{G} - \mathbf{x_i}^{G}) + F_i(\mathbf{x}_{\text{rand1,i}}{}^{G} - \mathbf{x}_{\text{rand2,i}}{}^{G}) \quad (7)$$

   b) Current-to-rand/2

$$\mathbf{v_i}^{G} = \mathbf{x_i}^{G} + (1 - F_i)(\mathbf{x}_{\text{rand1,i}}{}^{G} - \mathbf{x_i}^{G}) + F_i(\mathbf{x}_{\text{rand2,i}}{}^{G} - \mathbf{x}_{\text{rand3,i}}{}^{G}) \quad (8)$$

With so many parameters and strategies combinations, it is imperative for the algorithm to adaptively select the most suitable combination automatically during optimization. The self-adaptation mechanism of SaDE works by computing the probability of choosing a specific parameter or strategy based on its performance over the past iterations. The formula for computing the probability is as follows [20]:

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}}, \quad where\, S_{k,G} = \frac{\sum_{g=G-T}^{G-1} ns_{k,g}}{\sum_{g=G-T}^{G-1} ns_{k,g} + \sum_{g=G-T}^{G-1} nf_{k,g}} + \varepsilon \quad (9)$$
$$, (k = 1, 2, \ldots K; G > T)$$

where $ns_{k,G}$ and $nf_{k,G}$ are respectively the number of successful and the number of failed survival of newly generated trial vectors by parameter k, and $S_{k,G}$ is the ratio of both ns_k and nf_k accumulated over a learning period of T. $p_{k,G}$ is the probability of choosing parameter k for the next iteration G by each member of the population. This ensures that the probability of selecting a parameter which helped evolve fitter solutions is always kept updated throughout the course of optimization.

MMTS is the modified version of the original MTS which employs agents to execute local search strategies along the dimensions of its agents [23]. 3 local search strategies (LS) have been introduced in the original technique as follows:

1) LS 1: explore along one dimension from the first dimension to the last dimension of the agent.
2) LS 2: same as strategy 1 except that the search is performed only for ¼ of the dimensions.
3) LS 3: for each dimension, first evaluate three small steps from the current agent and heuristically determine the solution of the next step.

In both LS 1 and 2, the search range (SR) which determines the step size from the original dimension is halved (to a minimum of 10^-15) for every steps with unimproved fitness. Unlike the original MTS, MMTS compute the step sizes of dimensional search by first calculating the average of all mutual dimension-wise distances between current population members (AveDis) and then choosing one linearly reducing factors from among 5 ranges: [1, 0.02], [5, 0.02], [10, 0.02], [20, 0.02], [40, 0.02]. MMTS also does not pre-define the locations of all agents on a simulated orthogonal array (SOA), applying instead the niching technique clearing to ensure diversity between the fittest members in the local regions [25].

As the time allotted to the computation of the optima in each run is limited to a fixed number of functional evaluations (FEs), more FEs are allocated to the strategy that performs better for the specific problem. In the initial stage of a run, both SaEPSDE and MMTS are allowed to execute for equal amount of FEs. This number is adaptively reduced for the strategy that produced less successful offspring-parent replacements by computing the probabilities of selection in the manner described in (9). Figure 1 describes the procedure of SaEPSDE-MMTS.

B. Neighborhood niching

While SaEPSDE-MMTS is excellent in discovering single global optima even when the problem dimensions are scalable, it must be modified to suit dynamic optimization problems. As mentioned in Section I, population diversity has to be maintained throughout the run of a dynamic problem. Niching and clustering techniques have been demonstrated to be highly effective for such applications. To achieve this, we chose the neighborhood crowding technique [25]. De Jong (1975) introduced the concept of crowding from observing the competition within a population for resources [21]. In crowding, survival-of-fittest occurs within the niches of the most similar members of the population. Each newly-generated offspring competes not against its own parent, but its nearest neighbor. B. Qu et. al. introduced neighborhood crowding concept which modifies the mutation step size of evolving individuals as well [25].

Figure 1: SaEPSDE-MMTS

WHILE stopping criteria not met (FEs < Max FEs)
Step 1: Initialize a randomly distributed population **X** of size N within the range of [$\mathbf{X}^{\text{Lower}}$, $\mathbf{X}^{\text{Upper}}$]
Step 2: Randomly generate ensemble of parameters [F_i^G, C_i^G, $strat_i^G$, $CR\_method_i^G$] as described in Section IV.A.
Step 3: DO FOR i=1 to N
Evolve $\mathbf{x_i}$ in **X** based on specified parameters
Step 4: Evaluate trial vector $\mathbf{u_i}$
Step 5: Compare fitness of offspring with the most similar individual in **X**
IF $f(\mathbf{u}_i^G) \le f(\mathbf{x_d})$, where $\mathbf{d}$=min(dist($\mathbf{u_i}$,$\mathbf{x_k}$)), k=1,…N
$\mathbf{X}_\mathbf{d}^{G+1} = \mathbf{u}_i^G$
ENDIF
Step 6: Accumulate number of successful ns_k and failed nf_k replacements for each parameter in ensemble.
Step 7: Compute the probability p_k of selecting parameter k using eqn. (9)
IF Generation number G > learning period T
Update parameters for next iteration, [F_i^{G+1}, C_i^{G+1}, $strat_i^{G+1}$, $CR\_method_i^{G+1}$], choosing [F_k, C_k, $strat_k$, CR_method_k] if random number $rand_i < p_k$.
END IF
Step 9: Accumulate number of successful ns and failed nf improvement for SaEPSDE and MMTS.
Step 10: Compute the probability p_k of executing SaEPSDE and MMTS using (9)
IF Generation number G > learning period T
Update member v choice for [SaEPSDE, MMTS] in next iteration if random number $rand_i < p_v$.
END IF
Step 11: Increment generation number G= G+1
END WHILE

In neighborhood crowding, niches of a fixed population size are formed around local best members (*lbest*) such that a niche size of m would consist of the *lbest* and its nearest *m-1* neighbors. Neighborhood mutation is implemented with the SaEPSDE-MMTS by modifying the ensemble of strategies from two basic strategies DE/rand/1 (1) and DE/best/1 (2) to their neighborhood mutation forms:

$$\mathbf{v}_i^G = \mathbf{x}_{\text{rand\_n1},i}^G + F(\mathbf{x}_{\text{rand\_n2},i}^G - \mathbf{x}_{\text{rand\_n3},i}^G) \quad (10)$$

$$\mathbf{v}_i^G = \mathbf{x}_{\text{lbest},i}^G + F(\mathbf{x}_{\text{rand\_n1},i}^G - \mathbf{x}_{\text{rand\_n2},i}^G) \quad (11)$$

where $\mathbf{x}_{\text{rand\_n1}}$ to $\mathbf{x}_{\text{rand\_n3}}$ are random members chosen from the neighborhood of the local best $\mathbf{x}_{\text{lbest}}$. Neighborhood size m has been recommended as 1/25 or a minimum of 5 by Qu et. al. in their introduction of the technique [25]. The selection process for the SaEPSDE part of the algorithm is also modified from parent-offspring replacement to selection between the offspring and its nearest neighbor within the niche parent population. This process is explained in Figure 2:

Figure 2: Neighborhood mutation crowding in SaEPSDE

WHILE number of functional evaluations *FEs* < MaxFEs
Step 1: Initialize a randomly distributed population **X** of size *N*
Step 2: Compute Euclidean distance for all members in **X**
$\mathbf{dist}_{ij}=|X|=\sqrt{\sum(x_{i,j})^2}, j=1, \ldots D, i=1,2\ldots N$
Step 3: DO FOR *i*=1 to *N*
1) Sort neighboring individuals according to fitness
2) The fittest member is labeled as local best *lbest*
3) Sort individuals according to distance proximity to current *lbest*
4) Label *m-1* nearest individuals to current *lbest* as from the same niche
DO FOR *i*=1 to *N*
Step 4: Evaluate trial vector $\mathbf{u_i}$
Step 5: Compute the Euclidean distance between $\mathbf{u_i}$ and all individuals in **X**
Step 6: Compare fitness of offspring with the most similar individual in **X**
IF $f(\mathbf{u}_i^G) \le f(\mathbf{x_d})$, where $\mathbf{d}=\min(\text{dist}(\mathbf{u_i},\mathbf{x_k}))$, k=1,…*N*
$\mathbf{X_d}^{G+1}=\mathbf{u}_i^G$
END IF
Increment *FEs* = *FEs* + *N*
END WHILE

C. *Archive clearing and reinitialization*

With dynamic optimization, changes to the problem demand drastic changes in behavior for the algorithm. To check for dynamic changes over every generation, the current fittest member *gbest* and another randomly chosen member are re-evaluated for changes in their fitness to ensure that dynamic change has not occurred. If change is detected, the *gbest* and all local optima locations thus far discovered would be archived. This archive accumulates all solutions in the past, and is re-evaluated for every new search. Since simple archiving of past solutions may lead to repetitions, clearing is applied to this archive. An adaptive archive clearing procedure that had been previously proposed in [27] is applied here. The cleared archive would be used for the reinitialization of individuals due to dynamic change by replacing half of the new population with the fittest archive members and randomly reinitializing the rest.

V. EXPERIMENTAL RESULTS

The benchmark from IEEE WCCI 2014 Competition on Evolutionary Computation for Dynamic Optimization Problems comprised of the generalized dynamic benchmark problems (GDBP) [7]. The following are the functions and

TABLE I: ERROR VALUES FOR F1-F6

	Change ratio		Change Instance											
			T1	***T2***	***T3***	***T4***	***T5***	***T6***	***T7***	***T8.1***	***T8.2***	***T8.3***	***T9***	***T10***
F1	0.3	**Mean**	4.06e-03	2.95e-02	7.76e-02	1.89e-02	2.62e-02	7.81e-03					2.48e-02	7.88e-02
		Std. Dev	8.53e-03	5.44e-02	2.10e-02	2.64e-02	3.94e-02	9.52e-04					5.45e-02	3.78e-03
	0.7	**Mean**	8.55e-02	6.71e-01	7.68e-01	9.49e-02	2.22e-01	1.71e-01					8.58e-01	8.78e-01
		Std. Dev	1.23e-01	3.49e-01	2.00e-00	3.89e-01	2.43e-01	5.26e-01					9.46e-01	1.98e+00
	1	**Mean**	6.22e-02	4.92e-01	2.73e-01	1.85e-01	3.12e-01	2.16e-01	6.46e+01	8.18e-03	1.33e-01	3.08e+00	2.48e+00	6.98e-01
		Std. Dev	4.11e-01	5.66e-01	5.12e-00	3.14e-01	4.04e-01	9.20e-01	3.90e+01	4.77e-03	9.31e-01	4.27e+00	5.65e+00	3.55e+00
F2	**Change ratio**		**Change Instance**											
			T1	***T2***	***T3***	***T4***	***T5***	***T6***	***T7***	***T8.1***	***T8.2***	***T8.3***	***T9***	***T10***
	1	**Mean**	5.72e-01	9.12e+01	3.72e+01	2.84e+00	6.13e+01	2.16e+00	6.45e+00	7.15e-03	2.34e-01	9.80e+00	3.28e-01	3.96e-01
		Std. Dev	8.81e-01	7.65e+01	8.32e+01	2.34e-01	9.14e+00	9.33e-01	1.90e+01	4.47e-03	3.31e-01	4.97e+00	5.35e+00	5.51e+00
F3	**Change ratio**		**Change Instance**											
			T1	***T2***	***T3***	***T4***	***T5***	***T6***	***T7***	***T8.1***	***T8.2***	***T8.3***	***T9***	***T10***
	1	**Mean**	9.44e+02	4.32e+01	2.74e+02	1.35e+02	3.15e+02	7.16e+02	4.45e+01	4.28e+01	5.33e+01	9.18e+02	6.48e+02	5.97e+01
		Std. Dev	6.12e+02	2.61e+02	5.02e+02	3.14e+02	1.64e+02	8.20e+02	9.90e+02	4.57e+01	5.18e+01	1.22e+02	3.65e+02	3.75e+02
F4	**Change ratio**		**Change Instance**											
			T1	***T2***	***T3***	***T4***	***T5***	***T6***	***T7***	***T8.1***	***T8.2***	***T8.3***	***T9***	***T10***
	1	**Mean**	8.52e-01	6.41e+00	8.68e+00	5.49e+00	2.62e+00	1.11e+00	8.82e+00	5.28e-01	2.31e-01	2.98e+00	6.78e-01	9.28e-01
		Std. Dev	1.93e-01	2.49e+00	2.90e+01	5.59e+00	2.77e+00	2.26e+00	1.13e+00	4.97e-02	2.41e-01	7.67e-01	3.46e-01	1.88e+00
F5	**Change ratio**		**Change Instance**											
			T1	***T2***	***T3***	***T4***	***T5***	***T6***	***T7***	***T8.1***	***T8.2***	***T8.3***	***T9***	***T10***
	1	**Mean**	6.57e+00	4.92e+00	3.33e-01	1.25e+00	3.92e-01	3.16e-01	6.71e+00	7.78e-01	1.31e-00	2.88e+00	2.93e+00	7.98e+00
		Std. Dev	1.36e+00	3.61e+00	5.11e-00	9.16e+00	4.84e-01	1.27e-01	3.99e-01	4.79e-02	6.66e-01	1.27e+00	7.61e+00	3.50e+00
F6	**Change ratio**		**Change Instance**											
			T1	***T2***	***T3***	***T4***	***T5***	***T6***	***T7***	***T8.1***	***T8.2***	***T8.3***	***T9***	***T10***
	1	**Mean**	4.02e+01	4.05e+01	2.76e+01	6.82e+00	2.15e+01	2.19e+01	4.44e+00	4.16e+00	1.39e+01	3.89e+00	3.55e+01	1.58e+01
		Std. Dev	8.13e+01	5.67e+01	8.12e+00	3.94e+01	4.14e+01	9.30e+00	1.39e+01	4.57e+01	8.31e+01	2.27e+01	4.14e+01	3.58e+00

change types being tested:

- F_1 – Rotation peak function
- F_2 – Composition of Sphere's function
- F_3 – Composition of Rastrigin's function
- F_4 – Composition of Griewank's function
- F_5 – Composition of Ackley's function
- F_6 – Hybrid Composition function

Unlike the 2009 version of GDBP, another parameter known as the change ratio has been implemented to simulate partial change in the environment. With the exception of F1, a single run of each test require test function F2 - F6 to be executed over 60 change instances for 12 different change types (T1 – T10). For each function and each step changes, 20 separate runs have been executed and the accuracy recorded. Table 1 displays the mean and standard deviations of the absolute error in our tests.

VI. CONCLUSION

We have proposed a new niching DE variant with self-adaptive MMTS and neighborhood mutation for dynamic optimization problems (DOPs). The performance of our algorithm has been assessed based on the benchmark problems of the GDBG, and the promising results from our tests indicate the need for further development in our algorithm.

REFERENCES

[1] Y. Jin and J. Branke, "Evolutionary Optimization in Uncertain Environments – A Survey", IEEE Trans. on Evolutionary Computation, vol. 9, no. 3, pp. 303 – 317, June 2005.

[2] A. P. Engelbrecht, Computational Intelligence: An Introduction, 2nd ed. John Wiley & Sons, London, pp.70 – 85, 2007.

[3] Zhang JQ, Sanderson AC, "JADE: adaptive differential evolution with optional external archive.", IEEE Trans Evol Comput 13(5): 945–958

[4] K. Price, R. Storn, J. Lampinen., Differential Evolution: A practical approach to global optimization, 1st ed. Springer, Heidelberg, pp. 1-10, 135 – 185, June 2005.

[5] A. Qing, Differential Evolution: Fundamentals and Applications in Electrical Engineering, 1st ed., John Wiley & Sons, Singapore, pp. 1-30, 61-83, 2009.

[6] J. Branke, Evolutionary Optimization in Dynamic Environments, Norwell, MA: Kluwer, 2001.

[7] C. Li, M. Mavrovountiotis, S. Yang, X. Yao, "Benchmark Generator for the IEEE WCCI-2014 Competition on Dynamic Optimization: Dynamic rotation peak benchmark generator and Dynamic general benchmark generator", Technical Report, October 2013

[8] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems", Proc. of the 1999 Congress on Evolutionary Computation, pp. 1875-1882, 1999.

[9] R. W. Morrison and K. A. De Jong, "A test problem generator for non-stationary environments", Proc. of the 1999 Congress on Evolutionary Computation, pp. 2047-2053, 1999.

[10] S. Yang and C. Li, "A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments", IEEE Trans. on Evolutionary Computation, vol. 14, no. 6, pp 959-974, December 2010.

[11] E. L. Yu and P. N. Suganthan, "Evolutionary Programming with Ensemble of Explicit Memories for Dynamic Optimization", Proc. of the 2009 Congress on Evolutionary Computation, pp. 431 – 438, 2009.

[12] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec and V. Zumer, "Dynamic Optimization using Self-Adaptive Differential Evolution", Proc. of the 2009 Congress on Evolutionary Computation, pp. 415 – 422, 2009.

[13] T. Blackwell and J. Branke, "Multi-swarms, Exclusion and Anti-convergence in Dynamic Environments", IEEE Trans. on Evolutionary Computation, vol. 10, no. 4, pp 459-472, 2006.

[14] D. Parrott and X. Li, "Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation", IEEE Trans. on Evolutionary Computation, vol. 10, no. 4, pp 440-458, August 2006

[15] R. Mendes and A. S. Mohais, "DynDE: a Differential Evolution for Dynamic Optimization Problems", Proc. of the 2005 Congress on Evolutionary Computation, pp. 2808-2815, 2005.

[16] M. C. du Plessis and A. P. Engelbrecht, "Improved Differential Evolution for Dynamic Optimization Problems", Proc. of the 2008 Congress on Evolutionary Computation, pp. 229 – 234, 2008.

[17] R. Mallipeddi and P. N. Suganthan, "Differential Evolution Algorithm with Ensemble of Parameters and Mutation and Crossover Strategies", *Applied Soft Computing Journal*, v 11, n 2, p 1679-1696, March 2011.

[18] U. Halder, D. Maity, P. Dasgupta, S. Das, "Self-adaptive Cluster-Based Differential Evolution with an External Archive for Dynamic Optimization Problems", *Swarm, Evolutionary, and Memetic Computing*, Lecture Notes in Computer Science, Volume 7076, 2011, pp 19-26

[19] L. Liu, S. Yang, D. Wang, "Particle Swarm Optimization with Composite Particles in Dynamic Environments", IEEE Trans. on Systems, Man And Cybernetics – Part B: Cybernetics, vol. 40, no. 6, Dec 2010

[20] K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, 2005, pp. 1785-1791.

[21] S. Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large scale optimization," Soft Computing, vol. 43, no. 1, pp. 1-17, 2011.

[22] A. Pe´trowski, "A clearing procedure as a niching method for genetic algorithms." in Proceedings of the IEEE international conference on evolutionary computation, New York, USA, 1996, pp 798–803.

[23] L. Y. Tseng, C. Chen, "Multiple trajectory search for multiobjective optimization." in Proceeding 2007 IEEE congress on evolutionary computation, pp 3609–3616

[24] L. Y Tseng, C. Chen, "Multiple trajectory search for large scale global optimization." in Proceeding 2008 IEEE congress on evolutionary computation, pp 3052–3059

[25] B. Y. Qu, P. N. Suganthan, J. J. Liang, "Differential Evolution with Neighborhood Mutation for Multimodal Optimization", IEEE Transactions on Evolutionary Computation, v 16, n 5, p 601-14, Oct. 2012

[26] J. Derrac, S. García, S. Hui , F. Herrera, P. N. Suganthan, "Statistical Analysis of Convergence Performance Throughout the Evolutionary Search: A Case Study with SaDE-MMTS and Sa-EPSDE-MMTS.", 2013 IEEE Symposium on Differential Evolution (SDE), Singapore, pp. 151-156, April 16-19, 2013

[27] S. Hui, P. N. Suganthan, "Ensemble differential evolution with dynamic subpopulations and adaptive clearing for solving dynamic optimization problems 2012 IEEE Congress on Evolutionary Computation, CEC 2012