Dynamic Evolutionary Algorithm With Variable Relocation

Yonas G. Woldesenbet and Gary G. Yen, *Fellow, IEEE*

***Abstract*—Many real-world optimization problems have to be solved under the presence of uncertainties. A significant number of these uncertainty problems falls into the dynamic optimization category in which the fitness function varies through time. For this class of problems, an evolutionary algorithm is expected to perform satisfactorily in spite of different degrees and frequencies of change in the fitness landscape. In addition, the dynamic evolutionary algorithm should warrant an acceptable performance improvement to justify the additional computational cost. Effective reuse of previous evolutionary information is a must as it facilitates a faster convergence after a change has occurred. This paper proposes a new dynamic evolutionary algorithm that uses variable relocation to adapt already converged or currently evolving individuals to the new environmental condition. The proposed algorithm relocates those individuals based on their change in function value due to the change in the environment and the average sensitivities of their decision variables to the corresponding change in the objective space. The relocation occurs during the transient stage of the evolutionary process, and the algorithm reuses as much information as possible from the previous evolutionary history. As a result, the algorithm shows improved adaptation and convergence. The newly adapted population is shown to be fitter to the new environment than the original or most randomly generated population. The algorithm has been tested by several dynamic benchmark problems and has shown competitive results compared to some chosen state-of-the-art dynamic evolutionary approaches.**

***Index Terms*—Adaptation, dynamic evolutionary algorithm (DEA), optimization, relocation.**

I. Introduction

IN RECENT YEARS, evolutionary algorithms have been applied to real-world optimization problems under the presence of uncertainties with a mix of success. In general, there are four major categories of uncertainties that have been dealt with using evolutionary approaches: noise in the fitness function, perturbations in the design variables, approximation in the fitness function, and dynamism in optimal solutions [1]. Noise and approximation introduce uncertainty in the fitness function, while perturbation brings uncertainty in the decision space. On the other hand, dynamic optima result in uncertainty in the location, height, and width of the optimal solutions through time.

This paper focuses on dynamic optimization problems (DOPs), which are common uncertainty problems with possible changes in their objective function, constraints, environmental parameters, or problem representations during optimization [2]. These changes may reflect on the height, width, or location of the optimum solution or combination of these three [3]. A DOP can be formulated as

$$\text{Optimize} \quad f(X,e) = f(x_1, x_2, \ldots, x_n, e) \tag{1}$$

where each dimension of the search space is defined between $x_j^{\min} \leq x_j \leq x_j^{\max}$ for $j = 1, 2, \ldots, n$. f is the objective function to be optimized; $X = (x_1, x_2, \ldots, x_n)$ is the n-dimensional decision vector and e represents the environmental state whose variation can have either periodic or sporadic nature.

A good example is dynamic portfolio management, which is a common optimization problem in modern finance [4]. This problem aims to obtain an optimal allocation of assets that maximize profit, while minimizing risk of investment. In engineering, dynamic portfolio optimization problems are common in deregulated electricity markets in which the operations of different power stations are controlled and coordinated to maximize profit and minimize risk. There are various uncertainties in a deregulated electricity market, including spot market prices, load obligations, and strip/option prices [5]. The values for some of these factors change over time, and it is not unusual to optimize for the market price every hour.

Another scenario where DOPs arise comes from job shop scheduling problems. In these problems, new jobs may arrive or machines may break down or wear out during operations [6]. Therefore, the job schedules should be dynamically modified to accommodate the changes over time.

In general, a good dynamic evolutionary algorithm (DEA) should be able to track the changing optimal solution even under high severity and frequency of change. It must be able to reuse as much information as possible from previous generations to speedup the optimization search. Furthermore, the extra computational cost incurred should be reasonably comparable to its performance improvement.

The challenges in solving DOPs arise from the occurrence of changes in the location, number, and properties of the optimal solutions. When a standard evolutionary algorithm converges for a certain problem setting, the diversity and exploration capability of the population are greatly diminished. As a result, continuing the evolutionary process from the converged population without any adaptation scheme or facilitation of exploration leads to a higher probability of being unable to find the new optimal solutions or of being stuck in local optima. Therefore, it is necessary to implement certain schemes in the evolutionary algorithm to account for the dynamism of the optimization problem.

Manuscript received May 30, 2007; revised November 21, 2007 and March 15, 2008. First published February 27, 2009; current version published June 10, 2009.

The authors are with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: gyen@okstate.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2008.2009031

There are several important aspects of DOP and these include severity, frequency, observability, detectability, and dynamics of change [6]. While higher severity of change necessitates the DEA to increase diversity and exploration, higher frequency of change requires a faster convergence after a change has occurred. If the severity of change is too high for the DEA, the algorithm may not locate the new optima or might get stuck in local optima. Similarly, if the frequency of change is faster than the adaptation speed of the DEA, the algorithm will not reach the optimal solutions before another change emerges.

On the other hand, detection of change can be done in several ways assuming that the change is observable. Some of the common practices include checking if the time averaged best performance of DEA deteriorates and checking if the fitness of at least one of the reevaluated individuals has changed appreciably [6]. In most cases, however, researchers prefer to preset the change frequency in terms of the number of evaluations as in [6], and assume that the algorithm will detect the change explicitly. This assumption is vital because it allows the researchers to decouple the uncertainty handling and change detection problems and focus on a single issue. For the discussion of this paper, the authors have taken a similar assumption.

Furthermore, the DOP may involve different dynamics changes and these include constant, linear, circular/revolving, reshaping, and random modes [6]–[8]. In most cases, the algorithm is expected to work without knowing it explicitly. In this paper, we present a dynamic evolutionary algorithm that adapts individuals of the previous population when a change occurs based on evolutionary history. The adaptation is carried out by relocation that introduces shifts in the individuals' decision variables to enhance the population's diversity. Some sought qualities in dynamic evolutionary algorithms include reusability, faster convergence, higher accuracy, better adaptation, easier implementation, and improved performance. Reusability refers to the ability to reuse as much information as possible from the previous evolutionary process. Reusability normally provides faster convergence and allows the algorithm to adapt to the new environmental condition quickly. Higher severity of change reduces reusability of previous evolutionary data and demands greater exploration capability. Higher frequency of change demands faster convergence and adaptation to the new environment. Accuracy, on the other hand, refers to how close the best individual found is to the actual optimal solution. Improvement in the accuracy of the optimal solution may compromise the speed of convergence and hence, the algorithm should be equipped to balance between the additional computational cost and the observed performance improvement. Adaptation, alternatively, refers to adjusting the current population to the new environmental condition. Adaptation can be done by modifying the level of evolutionary operators, like mutation, to encourage exploration or by modifying the evolutionary process based on the previous evolutionary history.

In this paper, the above qualities are achieved by using variable relocation that adapts already converged or currently evolving individuals to the new environmental condition. The proposed algorithm relocates the individuals based on their change in functional value due to the change in the environment and the average sensitivities of their decision variables to the corresponding change in the objective space. The relocation radius introduces a certain amount of uncertainty to be applied to each individual and restores diversity and accelerates exploration. Furthermore, because the population is adapted from the previous population, there is a higher reuse of previous evolutionary data, which provides faster convergence. The relocation radius is specific to each individual, and this gives the algorithm better adaptation than those approaches that use a single adaptation value for the whole population. As a technique to be used at transient periods, the proposed algorithm provides the next evolutionary cycle with better initial population than a randomly generated population. As a result, there will be a considerable progress jump for the upcoming evolutionary process, and this gives the proposed algorithm faster adaptation and convergence.

This paper is structured as follows. Section II provides a brief overview of various evolutionary approaches proposed so far for DOPs. In Section III, the proposed variable relocation based dynamic evolutionary algorithm is elaborated and analyzed. After that in Section IV, different dynamic benchmark test problems are used to evaluate the proposed algorithm, and the results of the experiments are tabulated. Finally, we conclude this paper with a summary of current findings and ideas for future work.

II. Literature Survey

The following subsections provide a brief review of evolutionary approaches developed for DOPs. For more detailed reviews, the readers are referred to references cited in [1], [2], and [6].

A. Reinitialization

The most naïve approach ever conceived for solving DOPs is to reinitialize the evolutionary process when a change occurs. A similar approach restarts the population based on evolutionary algorithm convergence [9]. The deficiency of these approaches is that almost none of the past evolutionary materials are ever used, and this unavoidably hinders any possible speed up in convergence.

B. Memory-Based Approaches

Memory based approaches are commonly divided into two groups: *explicit* and *implicit* memory approaches. Explicit memory approaches are memory-based approaches that use external memory to store previous evolutionary materials that may be helpful in future stages of the evolutionary process. The common approaches under this category use a small sized memory to store the best solutions and add them back to the population if they are better fit than the current individuals [15]–[17]. In [15] and [16], Mori and colleagues solved DOPs using the thermodynamical genetic algorithm (TDGA), which evaluates the entropy of a population, and selects the population so as to minimize the free energy. They used archiving to allow utilization of individuals from the past adaptation as candidate solutions for recurrent changes in the landscape. In [19], Trojanowski and Michalewicz applied a short-term memory to recall some of the solutions of an individual's ancestors to increase the diversity by reintroducing individuals that have been considered good in recent generations. Finally, Bendtsen and Krink [20] used a dynamic memory model that is updated during the evolutionary process. They used the memory to store best individuals for each change period, but at the same time allowed the stored individuals to evolve by small amounts of Gaussian mutation in the direction of the current best individual.

Implicit memory approaches do not use an explicitly defined external memory, but some implicit form of memory exists in the system representation. One form of implicit memory is redundant representation, which is commonly used to slow down convergence and favor diversity. Diploidy is a common approach in redundant representations. In [21], [22], Smith and Goldberg developed a triallelic scheme where an allele can take one of the three values: "0," "recessive 1," or "dominant 1." In [23], Ng and Wong proposed using a diploid scheme with four possible alleles ("0 recessive," "1 recessive," "0 dominant," and "1 dominant"). In [24], a multilevel structured gene representation was used so that each level can activate or deactivate genes at the next lower level. In [25], the diploidy scheme proposed in [23] was further extended by a dominance change mechanism. In [26], an additive diploidy scheme was used where the genes determining one trait are added in order to determine the phenotypic trait. The phenotypic trait becomes 1 when a certain threshold is exceeded, and is 0 otherwise.

The basic assumption in memory approaches is that out of the stored information in the memory, there might be individuals that fall in the vicinity of the new optimal solution. This kind of assumption becomes inappropriate when there is noncyclic stochastic dynamism in the optimal solution. However, in principle, enhancing any of the uncertainty handling techniques by memory is a good practice.

C. Multiple Population Approaches

Multiple population approaches use several subpopulations to track multiple peaks in the landscape. A popular design proposed in [27], called shifting balance genetic algorithm, deployed one core population to exploit the best optimum found so far and several colonies to explore the search space. A diversity measure, distance to the core population, was included in fitness evaluations of the colonies. Another method proposed in [28], called self-organizing scouts, uses a small fraction of the population called "child population" to watch over the peaks, while the rest of the population searches for other peaks. The size of parent and child population is adaptively adjusted depending on the performance of the population. Another method proposed in [29], called multinationals genetic algorithm, used a "hill valley detection procedure" that defines the borders of the subpopulations. A valley is detected if the fitness in a sample point is lower than the fitness of both end points. Unfortunately, this method requires a large number of fitness evaluations. In [30], Blackwell and Branke proposed splitting a population of particles into a set of interacting swarms with local and global interaction for particle swarm optimization (PSO) algorithms. They tested their algorithm on multimodal dynamic moving peaks problems.

The challenge in this type of approach rests on how the algorithm should coordinate the operation of each subpopulation. As a result, these approaches tend to have large computational cost compared to single population approaches.

The multipopulation implementations used in this paper follow that of [6]. The first implementation is called P3, which is a standard evolutionary algorithm with three independent subpopulations. These populations independently explore the landscape and allow the population to retain a certain level of diversity when a change occurs. Similarly, P3/Mem is the variation of this implementation with a memory module. Another multipopulation implementation divides the population into memory and search subpopulations and is denoted as Mem/Search. While the main population evolves, a small part of the population is used to search for new peaks. Promising solutions are stored in memory during the evolutionary process for later retrieval. The search population allows constant search and maintains the level of diversity in the whole population.

D. Mutation and Self-Adaptation

When a change occurs, the population undergoes a transient state where the values of some evolution operators are changed so as to enhance diversity and performance. Cobb and Grefenstette [31] introduced the idea of hypermutation in which mutation probability is increased immediately after a change has occurred. In this method, the individuals undergo a drastic increase in the mutation level after a change, which improves the diversity of the population. Vavak and colleagues [32] introduced the idea of a variable local search that uses a step-by-step increase in the mutation level based on the performance of the population. In [8], different self-adaptation schemes were compared, including: uniform self-adaptation, different mutation level for each dimension, mutation with covariance matrix adaptation, and sphere mutation, which learns the upper and lower limits of the required mutation level. In [33], the multiplicative update rule is used to adapt the mutation level; while in [34], lognormal adaptation is used. Other suggested techniques include lifetime learning [35] and adaptive chaotic mutation [36].

E. Other Diversity Preserving Techniques

The basic idea behind maintaining diversity in DEAs is to prevent the algorithm from premature convergence. Any DEA designs that enhance the diversity, but do not fall into the above four categories, are broadly classified into this class. Grefenstette [10] proposed the idea of introducing randomly generated individuals at each generation. The introduction of random immigrants allows the algorithm to keep a certain level of diversity and exploration capability. The algorithm is easy to understand and implement, but provides little means of adapting the current individuals to the new environment. Furthermore, when the changes are severe, the algorithm requires a larger number of random immigrants, which inadvertently compromises the algorithm's performance.

Meanwhile, Andersen [11] approached the issue of diversity maintenance by using fitness sharing as a means to favor less populated areas. When a region is highly populated, fitness is shared by a large number of individuals, reducing or penalizing their fitness. On the other hand, if a region is less populated, the fitness is shared between few individuals and their fitness is less affected and unpenalized. As a result, less populated areas will be favored over highly populated areas, preserving diversity.

On the other hand, Ghosh and colleagues [12] used the age of an individual to favor the fitness of middle aged individuals, which will maintain the diversity of the overall population. Jin and coworkers [13] suggested imposing a lower threshold on step-size to maintain diversity in evolutionary strategies.

In [14], Bui *et al.* proposed using multiobjective optimization for dynamic environments by transforming a single objective DOP into a multiobjective optimization problem by adding an artificial objective function. The artificial objective function is intended to promote the diversity in the evolutionary process.

Although most researchers agree that having a diversified population is a good idea for DOPs, the fact remains that maintaining diversity may impose more computation time and may also slow down the evolutionary process.

In general, memory-based approaches are suitable for periodic optima; multipopulation approaches are tailored for competing peaks; mutation and self-adaptation techniques are appropriate for landscapes with very fast but less drastic changes; and maintaining diversity is specific for continuously moving optima [2].

The proposed algorithm is inspired by the comparatively small computational cost of self-adaptation schemes and also by the idea of having a better technique that can be used alongside other approaches for solving DOPs. In this spirit, it can be categorized under the self-adaptation scheme. The algorithm also utilizes a small memory to further improve its performance in cyclic changes. The unique attribute about this algorithm is that the self-adaptation scheme and the relocation radius are specific to each individual, and this allows the algorithm to provide an initial population better adapted to the new environment. In the next section, the proposed algorithm is presented.

III. Proposed Algorithm

The proposed dynamic evolutionary algorithm exploits evolutionary progress history to adapt the current population to the new environment. The algorithm introduces shifts in the decision space, the so-called relocations, which try to account for the estimated amount of change in the objective space. A complete description of the proposed algorithm is presented in detail next.

$f = f(X, e)$ represents the DOP we want to optimize; X represents the n-dimensional decision vector, and x_d denotes the dth-dimension decision variable. A nomenclature table is given in Appendix A for quick reference. For the discussion of this paper, the DOP is assumed to be a maximization problem. Note that a minimization problem can be converted into a maximization problem by multiplying with -1.

Δx_d denotes the child's evolutionary progress in the dth-dimension of the decision variable with respect to its parents. It is measured as the difference between the dth-dimension decision variable x_d of a child and that of the centroid of its parents. This can be mathematically expressed as

$$\Delta x_d(\text{child}) = x_d(\text{child}) - \frac{x_d(\text{parent1}) + x_d(\text{parent2})}{2}. \quad (2)$$

The evolutionary fitness progress, Δf, of a child with respect to its parents is measured as the difference between the fitness, f, of a child and the interpolated fitness of its parents. The interpolation is based on the distance between a child and its parents (ΔX_{C-P}). The farther a parent is from its child, the lesser is its contribution to the interpolated fitness, as shown in (3a)–(4) at the bottom of the page.

The average evolutionary progress in the dth-dimension decision variable $(\overline{\Delta x_d})$ of a child can be obtained as the weighted sum of the child's Δx_d and the average evolutionary progress in the dth-dimension decision variable $(\overline{\Delta x_d})$ of its parents. The same is true for the child's average evolutionary fitness progress $(\overline{\Delta f})$, except that we use the interpolated value of its parents' average fitness progress, as shown in (5) and (6) at the bottom of the next page, where $n\text{Gen}$ denotes the total number of generations either from the start of an evolutionary process or the last occurrence of change, whichever was more recent, up to the current generation. On the other hand, w represents the weight given to previous evolutionary progresses relative to the current one. Using $w = 1$, the weighted average is equivalent to taking a simple average of all the individual progresses (total sum divided by number of generation). This means that the effect of all the previous Δx_d and Δf values starting from the recent occurrence of change is accumulated and does not diminish through time. Using $w = 0$, the weighted average is just the current value and the previous Δx_d and Δf values have no effect. Using w

$$\begin{aligned}\Delta X_{C-P1} &= \|X(\text{child}) - X(\text{parent1})\| \\ &= \sqrt{\sum_{d=1}^{n} (x_d(\text{child}) - x_d(\text{parent1}))^2}, \end{aligned} \quad (3a)$$

$$\begin{aligned}\Delta X_{C-P2} &= \|X(\text{child}) - X(\text{parent2})\| \\ &= \sqrt{\sum_{d=1}^{n} (x_d(\text{child}) - x_d(\text{parent2}))^2}, \end{aligned} \quad (3b)$$

$$\begin{aligned}\Delta f(\text{child}) &= f(\text{child}) \\ &\quad - \left(\frac{\Delta X_{C-P2} \cdot f(\text{parent1}) + \Delta X_{C-P1} \cdot f(\text{parent2})}{\Delta X_{C-P1} + \Delta X_{C-P2}}\right) \end{aligned} \quad (4)$$

values between 0 and 1 implies that the weighted average gives more emphasis to the current value than those from the previous iterations. However, this is done in a way that allows the effect of the early Δx_d and Δf values to have less contribution in the average value and to have their effects diminished over time. In this paper, we used $w = 0.5$ so that we have an intuitively good balance between decay and persistence of the effect of the previous delta values

The average evolutionary progress in the decision space of an individual can then be obtained as

$$\overline{\Delta X} = \sqrt{\sum_{d=1}^{n} \left(\overline{\Delta x_d}\right)^2}. \tag{7}$$

The average sensitivity of the decision space to change in the objective space is defined as the ratio of the average evolutionary fitness progress to the average evolutionary progress in decision space. Mathematically

$$\overline{S^X} = \frac{\overline{\Delta f}}{\overline{\Delta X}}. \tag{8}$$

The average sensitivity of the dth-dimension of the decision space to change in the objective space can then be obtained as

$$\overline{s_d^x} = \overline{S^X} \cdot \frac{\overline{\Delta x_d}}{\overline{\Delta X}} = \overline{\Delta x_d} \cdot \frac{\overline{\Delta f}}{\sum\limits_{d=1}^{n} \left(\overline{\Delta x_d}\right)^2}. \tag{9}$$

In DOPs, the evolutionary fitness progress Δf can arise from changes in the decision space of an individual or changes in the environmental parameter. This can be approximately formulated as

$$\Delta f = \sum_{d=1}^{n} \left(\overline{s_d^x} \cdot \Delta x_d\right) + \overline{S^e} \cdot \Delta e \tag{10}$$

where $\overline{s_d^x}$ is the average sensitivity of the fitness to change in the dth-dimension of the decision space; $\overline{S^e}$ is the average sensitivity of the individual's fitness to change in the environment; Δx_d and Δe are the corresponding changes in the dth-dimension decision variable and the environmental parameter, respectively.

Under normal evolutionary process, the environmental parameter is constant, i.e., $\Delta e = 0$. Under such cases, equation (10) reduces to

$$\Delta f = \sum_{d=1}^{n} \left(\overline{s_d^x} \cdot \Delta x_d\right) = \overline{S^X} \cdot \Delta X. \tag{11}$$

The equality of the last and middle terms in (11) can be obtained by substituting (9) in the middle term and performing the summation.

Δe is different from zero during the transition period. However, if we reevaluate all of the previous individuals, then all the changes in the decision variables become zero ($\Delta x_d = 0$). In this case, (10) can be written as

$$\Delta f = f^{e2} - f^{e1} = \overline{S^e} \cdot \Delta e \tag{12}$$

where $f^{e2} - f^{e1}$ represents the difference between the functional values of an individual in the new (with superscript, $e2$) and old (with superscript $e1$) environments, respectively.

The proposed algorithm estimates the required offsets in the decision variables that will match the fitness changes caused by the environment. This is done through the relocation radius, which is the anticipated uncertainty in the decision space of an individual. The relocation radius is used to relocate all individuals with intent to restore or further enhance their fitness. It can be expressed as

$$\Delta R = \begin{cases} -\frac{f^{e2} - f^{e1}}{\overline{S^X}}, & f^{e2} \le f^{e1} \\ \min\left\{\frac{f^{e2}_{\text{best}} - f^{e1}}{\overline{S^X}}, \frac{f^{e2} - f^{e1}}{\overline{S^X}}\right\}, & f^{e2} > f^{e1} \end{cases} \tag{13}$$

where f^{e2}_{best} is the best fitness in the new environment.

The relocation offsets in each dimension of the decision space can then be obtained as

$$\Delta r_d = \frac{\Delta R \cdot \overline{s_d^x}}{\overline{S^X}} = \Delta R \cdot \frac{\overline{\Delta x_d}}{\overline{\Delta X}}. \tag{14}$$

If $|\Delta r_d| > x_d^{\max} - x_d^{\min}$, then Δr_d is trimmed to

$$\Delta r_d = \text{sign}(\Delta r_d) \cdot (x_d^{\max} - x_d^{\min}) \tag{15}$$

where $x_d^{\max}$ and $x_d^{\min}$ are the maximum and minimum limits of the dth-dimension decision variable, respectively; and sign (r) is a function that returns the sign of r.

On the other hand, if Δr_d is less than $\Delta r_d^{\min}$ (minimum allowable relocation offset in the dth-dimension decision variable), then

$$\Delta r_d = \Delta r_d^{\min}. \tag{16}$$

The variable $\Delta r_d^{\min}$ is varied based on the diversity of the population immediately before change. After validation of Δr_d, the relocation algorithm will generate a number of offspring as

$$x_d^{\text{new}} = x_d^{\text{old}} + p \cdot \Delta r_d \tag{17}$$

where p is a random number between 0 and 1.

$$\overline{\Delta x_d}(\text{child}) = \frac{\Delta x_d(\text{child}) + w \cdot n\text{Gen} \cdot \left(\frac{\overline{\Delta x_d}(\text{parent1}) + \overline{\Delta x_d}(\text{parent2})}{2}\right)}{w \cdot n\text{Gen} + 1}, \tag{5}$$

$$\overline{\Delta f}(\text{child}) = \frac{\Delta f(\text{child}) + w \cdot n\text{Gen} \cdot \left(\frac{\Delta X_{C-P2} \cdot \overline{\Delta f}(\text{parent1}) + \Delta X_{C-P1} \cdot \overline{\Delta f}(\text{parent2})}{\Delta X_{C-P1} + \Delta X_{C-P2}}\right)}{w \cdot n\text{Gen} + 1} \tag{6}$$

If the value of x_d^{new} lies outside the interval $[x_d^{\min}, x_d^{\max}]$, then the algorithm reassigns x_d^{new} as

if $x_d^{\mathrm{new}} > x_d^{\max}$,

$x_d^{\mathrm{new}} = x_d^{\max} + x_d^{\mathrm{old}} - x_d^{\mathrm{new}} = x_d^{\max} - p \cdot \Delta r_d$

else if $x_d^{\mathrm{new}} < x_d^{\min}$,

$x_d^{\mathrm{new}} = x_d^{\min} + x_d^{\mathrm{old}} - x_d^{\mathrm{new}} = x_d^{\min} - p \cdot \Delta r_d$

end

The individuals are relocated n_{reloc} times, and the best fit individual out of a parent and its offspring will be passed to the initial population of the new environment. This new initial population will be better adapted to the change and is claimed to converge quickly. After this initial population is obtained, the evolutionary process will proceed with its normal operation.

In general, a relocation radius lies between the minimum and maximum allowable shifts in the decision variables. Minimum relocation refers to cases where the fitness value is almost insensitive to changes in the environment. Maximum relocation, on the other hand, refers to cases where the fitness value is extremely sensitive to changes in the environment and corresponds to random initialization, because the designated relocation can pick up any of the allowable values in the decision space. Other intermediate values of relocation radius will try to introduce a certain amount of uncertainty over the decision space in which the proposed algorithm will look for better individuals. This way of relocation formulation will allow the algorithm to treat DOPs without regard to their dynamics of change. If the dynamics of change is homogenous, then the relocation radius of all individuals will have the same value. If the dynamics of change is heterogeneous and deterministic, the relocation radius will vary from individual to individual and the relocation amount will have a deterministic nature. Finally, if the dynamics of change have random nature, the proposed relocation scheme will account for the changes in the fitness landscape by taking average sensitivity values over the evolutionary run, and this will allow treating the changes stochastically.

Furthermore, because the population is adapted from the previous population, there is a higher reuse of previous evolutionary data, which provides faster convergence. The relocation radii are specific to each individual, and this gives the algorithm better adaptation than those approaches that use a single adaptation value for the whole population, such as [31]. As a technique to be incorporated during the transient periods, the proposed algorithm provides the next evolutionary cycle with better initial population than a randomly generated population. As a result, there will be a considerable progress jump for the upcoming evolutionary process, and this enables the proposed algorithm faster adaptation and convergence.

In addition to variable relocation, the algorithm uses a small archive to store the best individuals obtained so far. An optional amenity that improves the performance of the proposed algorithm is using clustering, which improves the steady state diversity. The need for clusters may arise due to the fact that even though the proposed algorithm provides a diversified and better adapted initial population for the new environment, it is mostly up to the evolutionary algorithm to further optimize the relocated individuals in the steady state, i.e., after relocation and until the next occurrence of change. For this reason, the final performance may not improve significantly in some cases despite evolving for a number of generations after relocation. For such cases, incorporating a clustering module in the proposed algorithm greatly helps the optimization process as relocation and clustering alternatively maintain the diversity of the population both in transient and steady states, respectively. The major problem with clustering is the additional computational cost for handling the various clusters. With this understanding, we run our simulation using two variations of the proposed algorithm. The first is RVDEA/Mem, which is a variable relocation-based dynamic evolutionary algorithm enhanced with a memory module. The individuals in memory are allowed to participate in the selection of parents for crossover. They will be added to the pool of possible parents during the selection process. When the memory module archive is full, the oldest individual will be replaced by the most recent one. The memory update does not occur at every generation, rather it occurs in certain intervals and when a change occurs. The individuals in memory can also be used to detect changes by checking if their fitness value has changed. This design will allow the algorithm to work even without explicit knowledge about the change. However, this way of change detection will not be pursued in the remainder of the paper so as to decouple change detection from uncertainty handling and to focus on a single issue. RVDEA/Mem is the basic core algorithm that the authors recommend to solve DOPs. But when better performance is required and steady state diversity plays a great role in the performance of the final solution, the authors recommend using the second version, which is RVDEA/Cluster. This version differs from RVDEA/Mem in the clustering module that provides superior performance at the cost of the increased computational cost. Multiple clusters are used during the evolutionary process to guide the selection and replacement procedures. These clusters improve the exploration capability of the algorithm and prevent the algorithm from being stuck in local minima. Each cluster allows the algorithm to explore different parts of the landscape and individuals to converge to different solutions. In our implementation, we used the "density-based" clustering algorithm proposed in [18]. This algorithm identifies clusters by connecting individuals if the distance, $\|x_i - x_j\|$, between them is lower than a given threshold value σ_{dist}. All interconnected groups of individuals, whose group size exceeds a specified minimum size are identified as a cluster. No individual is allowed to belong to more than one cluster. In case an individual does not belong to any of the clusters, it will be added to the closest cluster. These clusters obtained are then allowed to evolve separately. This will allow the algorithm to preserve steady-state diversity to a certain extent. To reduce the computation cost, clustering will be performed at certain intervals and after a change occurs.

The pseudocode for the proposed algorithm is given in Figs. 1 and 2.

A tournament selection is used to select individuals that will be reevaluated out of the main population and archive when a change occurs. The only modification to the selection process

```
Procedure RVDEA
/* RVDEA/Mem and RVDEA/Cluster */
begin
    k = 1
    nGen = 1
    Initialization
    Clear archive
    Evaluation
    while (not exit_condition) do
     begin
        Detection of change
        if change is detected
         begin
            Set change flag
            Transient_EA
            nGen = 1
         end
        else
         begin
            /*Clustering → *only for RVDEA/Cluster */
            Selection
            Recombination
            Mutation
            Evaluation
            Replacement
            k = k + 1
            nGen = nGen + 1
         end
     end
end
```

Fig. 1. Pseudocode for the proposed dynamic evolutionary algorithm using variable relocation.

```
Procedure Transient_EA
begin
    obtain average sensitivities to change of the decision variables in the landscape
    update archive (archive ← best individual)
    select and re-evaluate individuals out of the population and archive so that the total
      number of evaluations per generation remains constant
    obtain functional changes due to the environment for those individuals
    obtain relocation offsets for those individuals
    relocate those individuals $n_{reloc}$ times
    k = k + $n_{reloc}$ + 1
    select the best individual from a parent and its relocated offspring and place it in the
      initial population of the new environment
    reset $\overline{\Delta x_d}$ and $\overline{\Delta f}$ values for all individuals
    reset change flag
end
```

Fig. 2. Pseudocode for the transient evolutionary algorithm using variable relocation.

for reevaluation is that an already selected individual is not allowed to participate in the remaining tournaments for that given change. Other than that, the tournament selection used here is similar to that used in recombination and replacement.

The *exit_condition* in the pseudocode can be defined in several ways. Some of these are reaching maximum fitness evaluations, maximum generations, and maximum changes. On the other hand, the counter k in the pseudocode is used to keep track of the total generations. Similarly, nGen counts the generations, but it is restarted from one every time a change occurs. The *change* flag is set when change occurs and is reset after relocation. On the other hand, the clustering procedure is conducted only for RVDEA/Cluster. All the other procedures are similar for both RVDEA/Mem and RVDEA/Cluster.

IV. Experimental Results and Discussions

A. Experimental Settings

1) Test Problems: There are a number of dynamic benchmark problems suggested for testing DEAs. In this paper, we exploit six different test functions (MP1, DF2, DF3, DF4, DF5, DF6) as benchmark problems. Except for DF3, DF4, and DF5, the other problems are maximization problems. The problem formulations for these functions are given in Appendix B for the completeness of the presentation.

MP1 was introduced in [17] and its components are competing cones whose height, width, and location can be varied independently.

DF2 is the problem proposed in [37] and uses independently varying n-dimensional Gaussian peaks. Each peak's amplitude, center, and variance can be varied independently.

DF3, DF4, and DF5 are the moving parabola problems used in [33] and [34]. The dynamics of change may have different forms, including linear, random, and circular dynamics. In this paper, DF3 represents the linearly varying moving parabola problem; DF4 describes a randomly varying moving parabola problem, while DF5 represents a circularly varying moving parabola problem.

DF6 is a problem used in [6]. It has two landscapes with ten peaks each. The parameters of each peak can be varied independently.

2) Performance Measure: There are several performance indices that have been suggested to measure the performance of dynamic evolutionary algorithms. In this paper, we use offline error variation [38] as means to quantify the performance of the proposed algorithm.

Offline error variation index [38] is the most commonly used performance index in DEAs, and it is obtained as the average of the error between the true optimal fitness and the best fitness at each evaluation. It is mathematically expressed as

$$\overline{e_{\text{offline}}} = \frac{1}{T}\sum_{i=1}^{T}\left(f_{\text{true}} - f_{\text{best}}^{i}\right) \tag{18}$$

where i is the evaluation counter; T is the total number of evaluations considered; f_{true} is the true optimum solution that is updated whenever a change occurs; and finally f_{best}^{i} is the best individual out of the evaluations starting from the most recent occurrence of change until the current evaluation.

B. Experimental Results

The experiments were conducted on dynamic benchmark problems MP1 and DF2–DF6. Each test is run 50 times. We use a population size of 100, a crossover rate of 0.6, a mutation rate of 0.2, and a maximum fitness evaluations of 500 000 for all implementations. Each dimension in decision space is bound between 0 and 100. In addition, we use SBX crossover and mutation with distribution index of 0.7. Tournament selection is adopted in recombination and replacement schemes. In addition, it is also used in the selection of individuals for reevaluation. We used a tournament size of five. A single elitism is also used to preserve the performance of the best individual at a given generation.

For our experiment, we used two variations of the proposed algorithm, RVDEA-one enhanced with memory, **RVDEA/Mem**, and another using several clusters to preserve diversity, **RVDEA/Cluster**. All the extra evaluations that occur during reevaluation and relocation are accounted for so that the total number of evaluations will be the same with the other

TABLE I
DEFAULT VALUES FOR ADDITIONAL RVDEA PARAMETERS

Number of relocation (n_{reloc})	1, when change frequency = 2; 2, otherwise
Memory size	0, 4, 10, 16; Default = 10 [MP1] 10 [DF2-DF6]
Cluster minimum size (only for RVDEA/cluster)	Population size / 10 = 10
Cluster radius (only for RVDEA/cluster)	$\sqrt{\sum_{d=1}^{n} \frac{(x_d^{\max} - x_d^{\min})}{10}} \approx 20$

TABLE II
DEFAULT PARAMETER SETTINGS FOR DYNAMIC BENCHMARK PROBLEMS

Default number of peaks	10
Default change frequency	Every 50 generation (5,000 evaluations)
Peak shape	Cone [MP1] ,Gaussian [DF2], Parabola [DF3-DF5], Bell Curve [DF6]
Dimension	5, 10, 20, 50; Default = 5 [MP1] 5 [DF2-DF6]
Min. and Max. limit of each decision dimension	[0,100]
Height severity	7.0
Width severity	1.0
Min. and Max. peak height	[30,70]
Min. and Max. peak width	[1,12]
Peak shift length	1.0

algorithms. When a change occurs, only selected members of the population and memory are reevaluated so that the total number of evaluations per generation remains constant. These algorithms are compared at least once against the following approaches for solving DOPs. The first is standard evolutionary algorithm, which is denoted as **SEA**. A variation of this algorithm with a memory module is denoted as **SEA/Mem**. Another algorithm introduces a number of random immigrants when a change occurs [10]. In this paper, 25 random immigrants are migrated into the current population, as suggested in [6]. This implementation of random immigrants is accompanied by a memory module to enhance its performance. This implementation will be referred to as **RI25/Mem**. The number of offspring generated after a change is the population size minus the number of random immigrants, so that the total number of evaluations remains 100 per generation. The next algorithm for comparison in our experiment is self-organizing scouts [28], denoted as **SOS**. The other algorithm we tested is standard evolutionary algorithm with three independent subpopulations [6], denoted as P3. When this algorithm is enhanced with memory, it is denoted as **P3/Mem**. Another algorithm we used in our experiment divides the population into memory and search subpopulations [6], denoted as **Mem/Search**. In addition, we compared the proposed algorithm against a standard evolutionary algorithm with random immigrants and three independent subpopulations. This combination is denoted as **P3RI25**, and when it is enhanced with memory, it is denoted as **P3RI25/Mem**. The last algorithm we used in our experiments is the hypermutation algorithm [31]. We included a memory module in the implementation of this algorithm to enhance its performance. This implementation will be referred to as **HM/Mem**.

Table I provides the values used for the additional parameters in the proposed algorithm. As it can be seen from the table, RVDEA/Mem has two additional parameters other than the standard EA parameters. Similarly, RVDEA/Cluster has four additional parameters. The number of relocation is chosen so as to reduce the computational cost of the relocation algorithm and at the same time provide better initial population to the next evolutionary cycle. For our implementation, we found out that this balance occurs when relocation number equals two. Similarly, we chose the minimum cluster size to be 10 and cluster radius to be 20 since they provide a good result with a reasonable additional computational cost. These values also result in the number of clusters to be around 10.

Due to the lack of reported data for some of the algorithms under certain benchmark test functions, different test functions are compared against different sets of algorithms. MP1 test problem was tested on all of the listed algorithms cited earlier. On the other hand, DF6 was evaluated on SEA/Mem, RI25/Mem, P3/Mem, P3RI25/Mem, Mem/Search, HM/Mem, and the two variations of the proposed algorithm. All the other test functions were evaluated only on SEA/Mem, RI25/Mem, HM/Mem, RVDEA/Mem, and RVDEA/Cluster. The first test was conducted on MP1 with default parameter settings given in Table II. The values that apply were also used for the other test problems, unless stated otherwise. Furthermore, the number of evaluations is used as the primary counter for comparing all the algorithms. Note that the number of evaluations is the population size multiplied by the number of generations elapsed.

TABLE III
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF PEAK NUMBER ON MOVING CONE PEAKS BENCHMARK PROBLEM [MP1]

Peak no.	SEA	RI25/Mem [10]	HM/Mem [31]	P3 [6]	SOS [28]	RVDEA/ Mem	RVDEA/ Cluster
1	3.65	9.28	11.98	3.45	2.06	1.23	1.02
10	17.87	14.63	15.62	14.47	4.01	4.88	3.54
20	19.63	13.87	16.02	15.62	4.43	5.68	3.87
30	20.15	12.89	14.24	14.39	4.20	5.86	3.92
40	19.02	12.41	13.93	14.57	4.06	5.65	3.49
50	19.58	12.74	13.97	13.78	4.12	5.21	3.78
100	17.76	11.20	13.81	11.49	3.75	4.98	3.37
200	18.03	10.82	14.06	10.66	3.62	4.92	3.54

Results for P3 and SOS are as reported in [6].

Table III compares the performance of the proposed RVDEA with chosen state-of-the-art DEAs on MP1 based on the number of peaks. As can be seen from Table III, the proposed algorithm performs much better than the other DEAs, except SOS. For the case of SOS, the proposed algorithm with memory module performed better only for a single peak MP1 problem. For the other cases, RVDEA/Mem provided comparable or lesser results. As RVDEA is an adaptation scheme performed at the transient stage, other evolutionary techniques that enhance performance of algorithm, like diversity preservation, can be applied to further improve the steady-state performance of RVDEA, keeping the computational cost the same as SOS. Since the number of fitness evaluations is fixed when comparing algorithms, computational cost refers to the additional calculations required by the algorithm for its proper operation. For example, in the case of SOS, additional computation is needed for forming and organizing the scouts. On the contrary, the additional evaluations required by RVDEA are simple averaging operations that

TABLE IV
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF GENERATION BETWEEN CHANGES ON MOVING CONE PEAKS BENCHMARK PROBLEM WITH TEN PEAKS [MP1]

Gen no.	SEA	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	Mem/ Search [6]	SOS [28]	RVDEA/ Mem	RVDEA/ Cluster
2	24.53	25.19	21.34	20.12	18.74	15.62	15.82	12.91
5	22.43	22.12	19.13	19.15	14.54	8.59	8.89	7.67
10	21.05	20.80	18.78	18.84	11.95	6.51	7.21	6.048
25	19.10	19.77	16.70	17.10	9.41	4.93	5.35	4.28
50	18.28	17.87	14.63	15.62	7.74	4.01	4.88	3.54
100	17.32	17.03	13.79	14.23	6.58	3.62	4.12	3.14

Results for Mem/Search and SOS are as reported in [6].

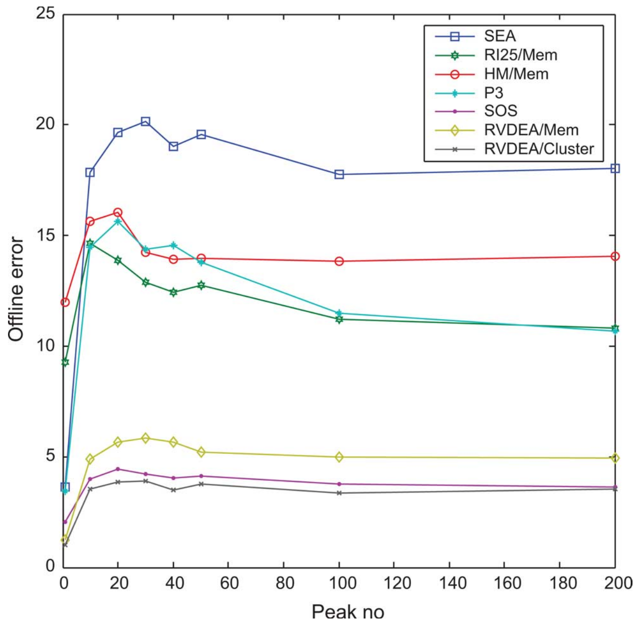

Fig. 3. Offline error versus peak number for selected DEAs on moving cone peaks benchmark problem [MP1].

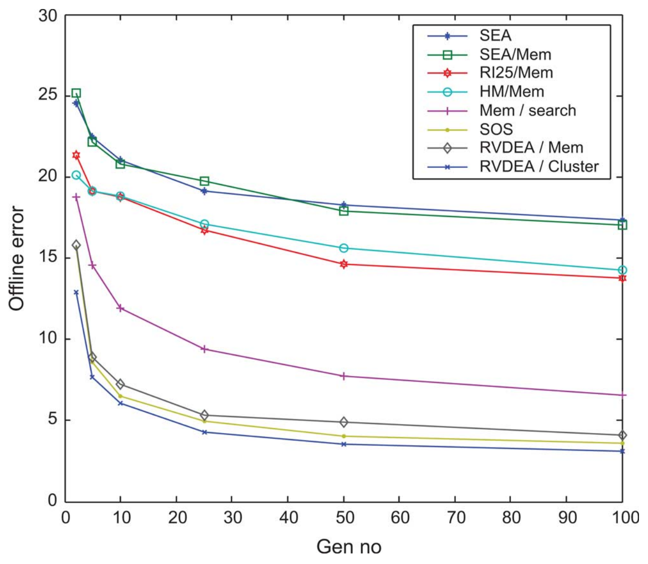

Fig. 4. Offline error versus number of generations between changes (change frequency) for selected DEAs on moving cone peaks benchmark problem [MP1].

are linearly dependent on the number of individuals in the population. This allows RVDEA to be used in conjunction with other techniques, such as steady-state diversity preservation to further enhance the algorithm's performance. In this notion, we tested a modified RVDEA with a clustering technique that has a comparable or smaller computational cost than SOS. In Table III, we see how well the modified algorithm's performance exceeded that of SOS in all different numbers of peak.

In Table IV, RVDEA was compared with the other algorithms based on frequency of change. RVDEA provides good results at a higher frequency of change and a comparable result as SOS when frequency is decreased. Due to the structure of the algorithm, the minimum allowable change frequency is two generations—one for reevaluation and one for relocation. Frequencies less than two generations can be analyzed by reducing the size of the original population so that RVDEA will have a minimum change frequency of at least two, while maintaining the number of evaluations constant.

Figs. 3 and 4 provide a graphical presentation of the offline errors of the different DEAs at varying peak numbers and varying frequency of change.

TABLE V
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF MEMORY SIZE ON MOVING CONE PEAKS BENCHMARK PROBLEM WITH TEN PEAKS [MP1]

Mem size	SEA/ Mem	RI25/ Mem [10]	HM/Mem [31]	P3/ Mem [6]	P3RI25/ Mem [6]	Mem/ Search	RVDEA/ Mem
0	20.18	15.83	16.89	n/a	n/a	n/a	4.90
4	17.92	13.21	14.06	14.53	17.88	7.76	5.15
10	17.87	14.63	15.62	14.45	18.73	7.34	4.88
16	17.66	13.61	13.77	14.49	21.21	7.46	5.06

Results for P3/Mem, P3RI25/Mem, and Mem/Search are reported in [6]. "n/a" stands for not available.

In Table V, we presented the performance of RVDEA as the memory size is changed. Although having memory is generally recommended in DEA, the algorithm still performs well without the support of memory.

In Table VI, we presented the performance of the different algorithms on MP1 test problem under different decision space dimensions. The authors believe that the MP1 test problem is a good representation of the DOPs and it will provide insight into the algorithm's performance under such conditions. As the dimension of the decision space increased, most of the algorithms

TABLE VI
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF DECISION SPACE DIMENSION ON MOVING CONE PEAKS BENCHMARK PROBLEM WITH TEN PEAKS [MP1]

Dimension	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	RVDEA/ Mem	RVDEA/ Cluster
5	17.78	14.63	15.62	4.88	3.54
10	22.94	18.12	19.86	5.86	4.64
20	28.15	22.16	25.74	7.34	6.86
50	34.02	26.78	29.94	10.63	9.92

TABLE VII
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF PEAK NUMBER ON TIME-VARYING GAUSSIAN PEAKS BENCHMARK PROBLEM [DF2]

Peak no.	SEA/ Mem	RI25/Mem [10]	HM/Mem [31]	RVDEA/ Mem	RVDEA/ Cluster
1	16.15	9.05	10.16	1.79	0.302
5	24.49	11.91	12.42	4.20	2.653
10	28.67	13.22	14.37	6.36	3.871
50	29.19	14.11	16.38	7.54	3.322
100	29.75	17.52	17.66	8.06	3.713
200	31.32	20.64	21.20	11.59	3.755

had difficulty in retaining their performance. The proposed algorithm, however, performed well even at higher decision space dimensions.

We also tested the algorithm using the time-varying Gaussian peaks problem [DF2]. We used the same setting as in Table II for this problem. The results were compared with the standard evolutionary algorithm with memory in Table VII. As can be clearly seen from the table, the proposed algorithm has better adaptation and performance even in problems with a higher number of peaks. RVDEA with memory provides good results, but further performance improvements can be obtained by using RVDEA with clusters. We also tested DF2 by varying the number of generations between changes. As can be observed from Table VIII, the proposed algorithm provides better results both at higher frequencies and lower frequencies of change compared to the chosen algorithms. As in the previous case, RVDEA with memory provides good performance, but RVDEA with cluster design provides far better results even though it exerts more computation. The algorithm is also tested for a moving parabola test problem with linear [DF3], random [DF4], and circular [DF5] dynamics. We run the problems with different cycle length. The results show that the proposed algorithm performs very well both in lower and higher frequencies of change. The results are summarized in Table IX. Furthermore, the algorithm performs well in problems with a higher number of peaks, as shown in Table X.

The proposed algorithm was also examined on the oscillating peaks function problem [DF6]. We used two landscapes with ten peaks each. The minimum and maximum peak widths parameters are set to 0.001 and 0.08, respectively. The rest of the parameters are kept the same as in Table II. In Table XI, the algorithm's performance under different cycle lengths is presented. As can be noticed from the results, the algorithm has better performance both in lower and higher frequencies of change, and the algorithm's performance was intact even with large variations in cycle lengths.

TABLE VIII
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF GENERATION BETWEEN CHANGES ON TIME-VARYING GAUSSIAN PEAKS BENCHMARK PROBLEM WITH TEN PEAKS [DF2]

Gen no.	SEA/Mem	RI25/Mem [10]	HM/Mem [31]	RVDEA/ Mem	RVDEA/ Cluster
2	25.10	21.15	20.76	14.41	10.11
5	32.42	17.86	19.24	8.83	7.55
10	31.78	15.99	16.28	6.98	4.41
25	29.80	14.60	15.56	6.44	4.12
50	28.67	13.22	14.37	6.36	3.87
100	25.19	11.53	12.43	5.95	3.34

TABLE IX
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF CYCLE LENGTH ON MOVING PARABOLA BENCHMARK PROBLEMS WITH TEN PEAKS [DF3-DF5]

Moving parabola Type	Cycle Length (eval)	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	RVDEA/ Mem	RVDEA/ Cluster
Linear [DF3]	1,000	10.893	8.70	8.83	2.334	0.881
	2,500	10.821	8.42	8.69	2.125	0.755
	5,000	10.865	8.31	8.54	2.082	0.609
	10,000	10.944	8.17	8.38	1.781	0.483
	20,000	11.291	8.08	8.22	1.622	0.299
	100,000	11.033	7.90	8.11	1.413	0.177
Random [DF4]	1,000	11.126	9.93	9.88	2.752	1.026
	2,500	11.065	9.76	9.78	2.611	0.982
	5,000	10.877	9.64	9.71	2.303	0.891
	10,000	10.531	9.42	9.59	2.142	0.769
	20,000	10.218	9.28	9.46	1.897	0.536
	100,000	9.893	9.17	9.33	1.662	0.247
Circular [DF5]	1,000	12.844	10.90	11.05	2.989	1.583
	2,500	12.815	10.75	10.96	2.788	1.457
	5,000	12.663	10.61	10.83	2.445	1.162
	10,000	12.587	10.47	10.72	2.121	0.783
	20,000	12.499	10.29	10.53	1.965	0.607
	100,000	12.431	10.22	10.41	1.792	0.340

TABLE X
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF PEAKS NUMBER ON MOVING PARABOLA BENCHMARK PROBLEMS WITH CYCLE LENGTH 5000 EVALUATIONS [DF3-DF5]

Moving parabola Type	Peaks no.	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	RVDEA/ Mem	RVDEA/ Cluster
Linear [DF3]	1	10.38	7.80	8.32	1.517	0.081
	5	10.67	7.97	8.41	1.892	1.122
	10	10.865	8.31	8.54	2.082	1.609
	50	14.45	8.64	8.69	2.367	1.756
	100	18.72	8.73	8.82	2.688	2.186
	200	21.08	8.88	9.02	2.850	2.377
Random [DF4]	1	10.31	9.21	9.19	1.268	0.106
	5	10.48	9.44	9.53	1.764	1.446
	10	10.877	9.64	9.71	2.303	1.791
	50	14.21	9.67	10.03	2.675	1.862
	100	18.64	10.32	10.55	2.904	1.985
	200	20.73	10.56	10.78	3.023	2.149
Circular [DF5]	1	12.30	9.86	10.08	1.687	0.158
	5	12.41	10.10	10.39	2.022	0.967
	10	12.663	10.61	10.83	2.445	1.162
	50	17.76	10.70	10.90	2.664	1.368
	100	21.19	10.92	10.95	2.864	1.743
	200	23.44	11.10	11.18	3.191	2.040

The effect of increasing the number of peaks on the performance of the algorithm more or less relates to an increase in problem difficulty and decrease in performance. But as can be seen from the results given in some of the tables, at some values of peak number there are discrepancies in the pattern. This is mainly due to the underlying benchmark. It is a well known effect that offline error decreases for a high number of peaks, because the maximization in the fitness function raises the average fitness of random individuals. On the other hand, as the number of generations between changes is increased, the performance

TABLE XI
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS AS FUNCTION OF CYCLE LENGTH ON OSCILLATING PEAKS BENCHMARK PROBLEM [DF6]

Cycle length (eval)	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	P3/ Mem [6]	P3RI25/ Mem [6]	Mem/ Search [6]	RVDEA/ Mem	RVDEA/ Cluster
1,000	11.91	9.00	10.74	10.07	9.87	7.19	4.252	2.648
5,000	11.47	8.52	9.31	n/a	n/a	n/a	4.132	2.411
10,000	14.13	7.85	8.09	n/a	n/a	n/a	3.978	2.342
100,000	17.40	7.23	7.76	12.00	9.27	4.71	3.821	2.025

Results for P3/Mem, P3RI25/Mem, and Mem/Search are as reported in [6].

TABLE XII
OFFLINE ERROR VARIATION AFTER 500 000 EVALUATIONS FOR VARIATIONS OF THE PROPSED ALGORITHM ON TEST PROBLEMS MP1 AND DF2-DF6 WITH TEN PEAKS

Algorithm	MP1	DF2	DF3	DF4	DF5	DF6
RVDEA without Memory	4.90	6.42	2.11	2.35	2.51	4.35
RVDEA/Mem with single relocation value	10.37	11.23	6.49	6.95	8.51	6.70
RVDEA/Mem	4.88	6.36	2.082	2.303	2.445	4.132
RVDEA/cluster	3.54	3.877	1.609	1.791	1.162	2.411

TABLE XIII
OFFLINE ERROR VARIATION AFTER 50 000 EVALUATIONS ON MOVING CONE PEAKS BENCHMARK PROBLEM [MP1]

Test problem	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	RVDEA/ Mem	RVDEA/ Cluster
DF1	19.05	15.61	15.06	4.94	3.63
DF2	25.29	12.86	16.53	6.28	3.82
DF3	10.02	10.09	8.31	2.12	0.712
DF4	11.64	11.41	10.71	2.28	0.884
DF5	12.15	10.33	12.83	2.41	1.17
DF6	14.67	10.75	8.87	4.13	2.41

TABLE XIV
THE P-VALUES FOR THE DISTRIBUTION OF OFFLINE ERROR VARIATION WHEN TESTED USING MANN–WHITNEY RANK-SUM TEST [39] ON MP1 BENCHMARK PROBLEM

	SEA/ Mem	RI25/ Mem [10]	HM/ Mem [31]	Mem/ Search [6]	SOS [16]
Offline Error (RVDEA/Mem, ---)	3.21e-5	3.21e-5	3.21e-5	8.26e-4	>0.05 no difference
Offline Error (RVDEA/Cluster, ---)	3.21e-5	3.21e-5	3.21e-5	3.21e-5	2.23e-3

of the algorithm generally deteriorates with increase in change frequency, with the exception of some discrepancies.

In Table XII, the proposed algorithm, RVDEA, is analyzed without memory (i.e., memory size set to zero) to see how much of the observed performance can be attributed to the proposed relocation mechanism in contrast to the use of memory or clusters. The results clearly suggest that variable relocation contributes to most of the observed performances. In addition, we tested the importance of using different relocation values for different individuals. To do this, we developed a modified version of RVDEA/Mem where all individuals are relocated based on a single value of relocation radius averaged over the population. The comparative result is presented in Table XII. The results clearly indicate that different individuals need different relocation after a change, and that this can be estimated by looking at the variable sensitivities of the ancestors.

In Table XIII, we presented the performance of the different algorithms when the maximum number of evaluations is set to 50 000. Both versions of the proposed algorithm were able to retain their superior performance at a reduced number of evaluations. The number of maximum evaluations does not have a predefined effect on the performance. While in some cases the performance increased, in other cases it decreased. This is because in DOPs, the number of evaluations before change is more important than the maximum number of evaluations.

We also tested the significance of the difference between the algorithms using the Mann–Whitney rank sum test [39]. In Table XIV, we presented the p-values with respect to the alternative hypothesis (i.e., $\mathrm{p} - \mathrm{value} < \alpha = 0.05$) for each pair of the proposed algorithm and a selected DEA. The distribution of the proposed algorithm has significant differences than those selected DEA, unless stated. As it can be seen from the table, RVDEA/Mem performs significantly better than most algorithms except SOS, while RVDEA/Cluster performs significantly better than all the algorithms for MP1 test problem. For the other test problems, the performance of RVDEA/Mem and RVDEA/Cluster is significantly higher, and better performance can be easily claimed without the rank sum test.

Overall, the experimental results show that RVDEA is capable of solving DOPs. The proposed algorithm is very easy to implement and the additional computational cost is very low. The proposed algorithm uses the evolutionary progress of each individual to arrive at the optimal relocation needed specifically for that individual to adapt to the new environment. Hence, it is a fast and effective adaptation scheme that occurs during the transient stages of a change. Furthermore, because the algorithm is exceptionally fast and simple, additional enhancements in population diversity can be easily incorporated at each steady state operation without bypassing the acceptable computational cost limit. In this paper, we implemented multiple clusters as a means to maintain steady state diversity. By using this simple clustering scheme, the algorithm performed better than all other DEAs tested in this paper. This performance improvement shows that RVDEA has a great potential to be used alongside other dynamic evolutionary techniques. Furthermore, the ease of implementing the algorithm makes the proposed algorithm very attractive. It should be noted that the algorithm still performs better than most of the other algorithms cited in this paper, even without any performance enhancement techniques.

V. CONCLUSION

Many real-world optimization problems have to be performed under the presence of various uncertainties. A significant number of uncertainty problems fall into the dynamic optimization category, where the fitness landscape undergoes various changes during the optimization. This paper proposes a

DEA that uses variable relocation to adapt already converged or currently evolving individuals to the changing landscape. The proposed algorithm relocates the individuals based on their change in function value due to the change in the environment and the average sensitivities of their decision variables to the corresponding change in the objective space. The relocation radius introduces a certain amount of uncertainty to be applied specifically to each individual and, in effect, restores diversity and accelerates exploration. Because the adaptation is conducted on the previous population, the proposed algorithm provides higher reusability of previous evolutionary data. Furthermore, the algorithm provides faster convergence and better adaptation, which makes it attractive for optimizing fitness landscapes with higher frequency of change. In addition, the algorithm is able to find optimal solutions in higher severities of change. Severe changes require higher diversity restoration and the proposed algorithm uses a larger relocation radius to do so. The relocated population is shown to be a better fit to the new environment than the original or a randomly generated population. The algorithm has been tested on several dynamic benchmark problems and has shown better results compared to some chosen state-of-the-art dynamic evolutionary approaches.

The relocation radius is specific to each individual, which gives the algorithm better adaptation than those approaches that use a single adaptation value for the whole population. Furthermore, using specific sensitivities and variable relocation allows the algorithm to provide a considerable progress jump for the next evolutionary process. As a technique to be used at transient periods, the proposed algorithm provides the next evolutionary cycle with better initial population than any other randomly generated population. Furthermore, the extra computational cost of the proposed algorithm is comparable to its performance improvement because the additional calculations incurred are basic arithmetic operations. The algorithm can be easily integrated into standard evolutionary algorithms and other uncertainty handling techniques, such as multipopulation and diversity preservation. This fast adaptation scheme, when enhanced with diversity preservation techniques, provides better overall performance. The authors believe that better results can also be obtained by implementing the relocation scheme on multipopulation approaches. For future work, the authors recommend applying the variable relocation scheme as a better fit to the multipopulation approaches for solving dynamic optimization problems.

APPENDIX A
NOMENCLATURE TABLE

X	Decision variable .
f	Fitness function.
x_d	dth-dimension decision variable.
Δx_d	Evolutionary progress or change in the dth-dimension decision variable.
ΔX_{C-P}	Distance between a child and its parent; index 1 and 2 are used to signify first and second parent.
$\overline{\Delta x_d}$	Average evolutionary progress in the dth-dimension decision variable.
Δf	Evolutionary progress or change in fitness function.
$\overline{\Delta f}$	Average evolutionary progress in fitness function.
$\overline{\Delta X}$	Average evolutionary progress in the decision space.
$\overline{S^X}$	Average sensitivity of the decision space to change in the objective space.
$\overline{s_d^x}$	Average sensitivity of the dth-dimension of the decision space to change in the objective space
Δe	Change in the environment.
ΔR	Relocation radius.
Δr_d	Relocation offsets in dth-dimension of the decision space.
x_d^{new}	dth-dimension decision variable after relocation.
x_d^{old}	dth-dimension decision variable before relocation.

APPENDIX B
DYNAMIC BENCHMARK TEST PROBLEMS

There are a number of dynamic benchmark problems suggested for testing DEAs. In this paper, we used the following test functions.

MP1 Moving Cone Peaks Benchmark Problem [17]

$$F(\vec{x},t) = \max(B(\vec{x}), \max_{i=1,\ldots,M} P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t)))$$

where $B(\vec{x})$ is a time-invariant "basis" landscape and P is the function defining the cone-shaped peaks , where each of the m peaks has its own time-varying parameters: height (h), width (w), and location $(\vec{p}_i(t))$.

DF2 Time-Varying Gaussian Peaks Problem [37]

$$f(X,t) = \max_{i=1,N}\left[A_i(t)\text{Exp}\left(\frac{-d(X,C_i(t))^2}{2\sigma_i^2(t)}\right)\right]$$

where $A_i(t)$ is the amplitude, $C_i(t)$ is the center and $\sigma_i(t)$ represents the width of the n-dimensional Gaussian peak.

DF3, DF4, and DF5 Moving Parabola Problems [33], [34]

$$f(x,t) = \min\sum_{i=1}^{n}(x_i+\delta_i(t))^2.$$

DF3 Linear Translation

$$\delta_i(0) = 0 \quad \forall i \in \{1,\ldots,n\}$$
$$\delta_i(t) = \delta_i(t-1) + s.$$

DF4 Random Dynamics

$$\delta_i(0) = 0 \quad \forall i \in \{1, \ldots, n\}$$
$$\delta_i(t) = \delta_i(t-1) + s \times N_i(0,1).$$

DF5 Circular Dynamics

$$\delta_i(0) = \begin{cases} 0 & : i \quad \text{odd} \\ s & : i \quad \text{even} \end{cases}$$
$$\delta_i(t) = \delta_i(t-1) + s \times c(i,t)$$
$$\text{where}$$
$$c(i,t) = \begin{cases} \sin\left(\frac{2.\pi.t}{\gamma}\right) & : i \quad \text{odd} \\ \cos\left(\frac{2.\pi.t}{\gamma}\right) & : i \quad \text{even.} \end{cases}$$

DF6 Oscillating Peaks Function [6]

$$f_i(t) = \min w(t) f_i(0)$$
$$w(t) = \frac{1}{3}\cos\left(\frac{2t\pi}{steps} + 2\pi\frac{i-1}{l}\right) + \frac{2}{3}, \quad i = 1, \ldots, l$$

where l is number of landscapes and $steps = 10$.

References

[1] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—A survey," *IEEE Trans. Evol. Comput.*, vol. 9, pp. 303–317, 2005.

[2] Y. Jin, "Evolutionary computation in dynamic and uncertain environments," in *IEEE Congress on Evolutionary Computation Tutorial, Honda Research Institute Europe*, Germany, 2004.

[3] R. W. Morrison and K. A. de Jong, "A test problem generator for non-stationary environments," in *Proc. Congr. Evol. Comput.*, Washington, DC, 1999, pp. 2047–2053.

[4] R. Merton, *Continuous-Time Finance*. Oxford, U.K.: Basil Blackwell, 1990.

[5] J. Xu, P. B. Luh, F. B. White, E. Ni, and K. Kasiviswanathan, "Power portfolio optimization in deregulated electricity markets with risk management," *IEEE Trans. Power Syst.*, vol. 21, pp. 145–158, 2006.

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

[7] S. Yang, "Constructing dynamic test environments for genetic algorithms based on problem difficulty," in *Proc. Congr. Evol. Comput.*, San Diego, CA, 2004, pp. 1262–1269.

[8] K. Weicker and N. Weicker, "Dynamic rotation and partial visibility," in *Proc. Congr. Evol. Comput.*, San Diego, CA, 2000, pp. 1125–1131.

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

[10] J. J. Grefenstette, "Genetic algorithms for changing environments," in *Proc. Parallel Problem Solving from Nature*, Brussels, Belgium, 1992, pp. 137–144.

[11] H. C. Andersen, *An Investigation into Genetic Algorithms, and the Relationship Between Speciation and the Tracking of Optima in Dynamic Functions*. Brisbane, Australia: Honors, Queensland Univ. Technol., 1991.

[12] A. Ghosh, S. Tsutsui, and H. Tanaka, "Function optimization in non-stationary environment using steady state genetic algorithms with aging of individuals," in *Proc. Congr. Evol. Comput.*, Anchorage, AK, 1998, pp. 666–671.

[13] Y. Jin and B. Sendhoff, "Constructing dynamic test problems using the multi-objective optimization concept," in *Applications of Evolutionary Computing*, G. Raidl, Ed. *et al.* New York: Springer, 2004, vol. 3005, pp. 525–536.

[14] L. T. Bui, J. Branke, and H. A. Abbass, "Multiobjective optimization for dynamic environments," in *Proc. Congr. Evol. Comput.*, Edinburgh, U.K., 2005, pp. 2349–2356.

[15] N. Mori, H. Kita, and Y. Nishikawa, "Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm," in *Proc. Parallel Problem Solving from Nature*, Amsterdam, The Netherlands, 1998, pp. 149–158.

[16] N. Mori, H. Kita, and Y. Nishikawa, "Adaptation to a changing environment by means of the thermodynamical genetic algorithm," *Trans. Inst. Syst., Control, Inf. Eng.*, vol. 12, no. 4, pp. 240–249, 1999.

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

[18] M. Ester, H. P. Kriegel, J. Sander, and X. Xiaowei, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discovery and Data Mining*, Portland, OR, 1996, pp. 226–231.

[19] K. Trojanowski and Z. Michalewicz, "Searching for optima in non-stationary environments," in *Proc. Congr. Evol. Comput.*, Washington, DC, 1999, pp. 1843–1850.

[20] C. N. Bendtsen and T. Krink, "Dynamic memory model for non-stationary optimization," in *Proc. Congr. Evol. Comput.*, Honolulu, HI, 2002, pp. 145–150.

[21] R. E. Smith, "Diploid genetic algorithms for search in time varying environments," in *Proc. Annu. Southeast Reg. Conf. ACM*, Birmingham, AL, 1987, pp. 175–179.

[22] D. E. Goldberg and R. E. Smith, "Non-stationary function optimization using genetic algorithms with dominance and diploidy," in *Proc. Int. Conf. Genetic Algorithms*, Cambridge, MA, 1987, pp. 59–68.

[23] K. P. Ng and K. C. Wong, "A new diploid scheme and dominance change mechanism for non-stationary function optimization," in *Proc. Int. Conf. Genetic Algorithms*, Pittsburgh, PA, 1995, pp. 159–166.

[24] D. Dasgupta and D. R. McGregor, "Non-stationary function optimization using the structured genetic algorithm," in *Proc. Parallel Problem Solving From Nature*, Brussels, Belgium, 1992, pp. 145–154.

[25] J. Lewis, E. Hart, and G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems," in *Proc. Parallel Problem Solving From Nature*, Amsterdam, The Netherlands, 1998, pp. 139–148.

[26] C. Ryan, "Diploidy without dominance," in *Proc. Nordic Workshop on Genetic Algorithms*, Helsinki, Finland, 1997, pp. 63–70.

[27] F. Oppacher and M. Wineberg, "The shifting balance genetic algorithm: Improving the GA in a dynamic environment," in *Proc. Genetic Evol. Comput. Conf.*, Orlando, FL, 1999, pp. 504–510.

[28] J. Branke, T. Kaubler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Proc. Adaptive Comput. Conf. Design Manuf.*, Berlin, Germany, 2000, pp. 96–101.

[29] R. K. Ursem, "Multi-national GA optimization techniques in dynamic environments," in *Proc. Genetic Evol. Comput. Conf.*, Las Vegas, NV, 2000, pp. 19–26.

[30] T. Blackwell and J. Branke, "Multi-swarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, pp. 459–472, 2006.

[31] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *Proc. Int. Conf. Genetic Algorithms*, Urbana–Champaign, IL, 1993, pp. 523–530.

[32] F. Vavak, T. C. Fogarty, and K. Jukes, "A genetic algorithm with variable range of local search for tracking changing environments," in *Proc. Parallel Problem Solving From Nature*, Berlin, Germany, 1996, pp. 1141–1146.

[33] P. J. Angeline, "Tracking extrema in dynamic environments," in *Proc. Int. Conf. Evol. Program.*, Indianapolis, IN, 1997, pp. 1213–1219.

[34] T. Back, "On the behavior of evolutionary algorithms in dynamic environments," in *Proc. Congr. Evol. Comput.*, Anchorage, AK, 1998, pp. 446–451.

[35] T. Sasaki and M. Tokoro, "Adaptation under changing environments with various rates of inheritance of acquired characters," in *Proc. Symp. Artif. Life Robot.*, Tokyo, Japan, 1998, pp. 34–41.

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

[37] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: A genetic algorithm approach," in *Proc. Congr. Evol. Comput.*, Washington, DC, 1999, pp. 2031–2038.

[38] K. de Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975.

[39] J. D. Knowles, L. Thiele, and E. Zitzler, A tutorial on performance assessment of stochastic multiobjective optimizers Computer Engineering and Network Laboratory, ETH, Zurich, Switzerland, TIK-Report 214, 2006.

Yonas Gebre Woldesenbet received the B.S. degree in electrical engineering from Bahir Dar University, Ethiopia, in 2004 and the M.S. degree in electrical engineering from Oklahoma State University, Stillwater, in 2007, and received the Gold-Medal Award for the most outstanding academic achievement in Bahir Dar University.

He is currently a Software Engineer. His research interest comprises of a wide variety of subject matters including dynamic evolutionary optimization, constraint handling in multiobjective optimization, intelligent systems, VLSI, and mathematical modeling and analysis.

Gary G. Yen (S'87–M'88–SM'97–F'09) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, in 1992.

He is currently a Professor at the School of Electrical and Computer Engineering, Oklahoma State University (OSU), Stillwater. Before joining OSU in 1997, he was with the Structure Control Division, U.S. Air Force Research Laboratory, Albuquerque, NM. His research is supported by the DoD, DoE, EPA, NASA, NSF, and Process Industry. His research interest includes intelligent control, computational intelligence, conditional health monitoring, signal processing and their industrial/defense applications.

Dr. Yen was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, the *IEEE Control Systems Magazine*, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, *Automatica*, and the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, PART A and PART B. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION AND MECHATRONICS. He served as the General Chair for the 2003 IEEE International Symposium on Intelligent Control held in Houston, TX, and the 2006 IEEE World Congress on Computational Intelligence held in Vancouver, Canada. In addition, he served as Vice President for the Technical Activities of the IEEE Computational Intelligence Society in 2005 and 2006, and is the founding Editor-in-Chief of the *IEEE Computational Intelligence Magazine* since 2006.