Chapter 2
Evolutionary Dynamic Optimization: Methodologies

Trung Thanh Nguyen, Shengxiang Yang, Juergen Branke, and Xin Yao

Abstract. In recent years, Evolutionary Dynamic Optimization (EDO) has attracted a lot of research effort and has become one of the most active research areas in evolutionary computation (EC) in terms of the number of activities and publications. This chapter provides a summary of main EDO approaches in solving DOPs. The strength and weakness of each approach and their suitability for different types of DOPs are discussed. Current gaps, challenging issues and future directions regarding EDO methodolgies are also presented.

2.1 Introduction

Many real-world objects are changing over time. For example, people are aging, the climate is changing, the stock market is moving up and down, and so on. As a result, it is important to be able to optimize in a dynamic environment. Changes may affect

Trung Thanh Nguyen
School of Engineering, Technology and Maritime Operations,
Liverpool John Moores University, Liverpool L3 3AF, U.K.
e-mail: T.T.Nguyen@ljmu.ac.uk

Shengxiang Yang
Centre for Computational Intelligence (CCI), School of Computer Science and Informatics,
De Montford University, The Gateway, Leicester LE1 9BH, U.K.
e-mail: syang@dmu.ac.uk

Juergen Branke
Warwick Business School, University of Warwick, Coventry CV4 7AL, U.K.
e-mail: juergen.branke@wbs.ac.uk

Xin Yao
Centre of Excellence for Research in Computational Intelligence and Applications
(CERCIA), School of Computer Science, University of Birmingham,
Birmingham B15 2TT, U.K.
e-mail: x.yao@cs.bham.ac.uk

S. Yang and X. Yao (Eds.): *Evolutionary Computation for DOPs*, SCI 490, pp. 39–64.
DOI: 10.1007/978-3-642-38416-5_2

the objective function, the problem instance, and/or the constraints [16, 124]. Hence, the optimal solution(s) of the problem being considered may change over time.

Formally, a dynamic optimization problem can be defined as follows [71]:

Definition 2.1 (Dynamic optimization problem) *Given a time-dependent problem f_t, an optimization algorithm G to solve f_t, and a given optimization period $\left[t^{begin}, t^{end}\right]$, f_t is called a **dynamic optimization problem** in the period $\left[t^{begin}, t^{end}\right]$ if during $\left[t^{begin}, t^{end}\right]$ the underlying fitness landscape that G uses to represent f_t changes **and** G has to react to this change by providing new optimal solutions*[1].

Evolutionary computation (EC) methods are good tools to solve DOPs due to their inspiration from natural systems, which have always been subject to changing environments. The study of applying evolutionary algorithms (EAs) and similar techniques to solving DOPs is termed *evolutionary optimization in dynamic environments* or *evolutionary dynamic optimization* (EDO) in this chapter. Over the last twenty years, a great number of different EDO methodologies have been proposed. The purpose of this chapter is to provide a summary of main EDO approaches in solving DOPs[2]. In-depth discussion of the strength and weakness of each approach will be provided, plus the suitability of each approach for different types of DOPs. Some future research issues and directions regarding EDO will also be presented.

The rest of this chapter is organized as follows. Section 2.2 reviews different approaches that have been developed by researchers to address DOPs. The strength and weakness of different approaches are also discussed there. Section 2.3 presents theoretical development regarding EDO methodologies. Finally, Section 2.4 summarizes the chapter and presents some discussions on current gaps, challenging research issues, and future directions regarding EDO methodologies.

2.2 Optimization Approaches

2.2.1 The Goals of EDO Algorithms

In stationary optimization, in most cases the only goal of optimization algorithms is to find the global optimum as fast as possible. However, in current EDO research where the considered problems are time-varying, the goal of an algorithm turns from finding the global optimum to firstly detecting the changes and secondly tracking the changing optima (local optima or ideally the global optimum) over time. In addition, in case the problem-after-change somehow correlates with the problem-before-change, an optimization algorithm also needs to learn from its previous search

[1] A more detailed version of this definition for DOPs was provided in [70, Chapter 4] and [74].

[2] An broader literature review, which is extended from this chapter, covering not only methodologies but also other aspects in EDO, can be found in [71].

experience as much as possible to hopefully advance the search more effectively. Otherwise, the optimization process after each change will simply become the process of solving a different problem starting with the old population/structure.

The following sections will briefly review typical approaches in EDO that have been proposed to satisfy the goals above. We will discuss the strength and weakness of the approaches and their suitability for different types of problems.

2.2.2 Detecting Changes

Many EDO approaches take an explicit action to respond to a change in the environment. This either assumes that changes in the environment are made known to the algorithm, or that the algorithm has to detect the change. If algorithms have to detect changes, they generally follow one of the following approaches: (a) detecting changes by re-evaluating dedicated detectors, or (b) detecting changes based on algorithm behaviors.

2.2.2.1 Detecting Changes by Re-evaluating Solutions

Overview

By far the most common change-detection approach is re-evaluating existing solutions. The algorithm regularly re-evaluates some specific solutions (detectors) to detect changes in their function values and/or feasibility. Detectors can be a part of the population, such as the current best solutions [48, 54, 58, 70], a memory-based sub-population [15, 130], or a feasible sub-population [70, 75]. Detectors can also be maintained separately from the search population. In this case, they can be just a fixed point [25], one or a set of random solutions [26, 82, 98], a regular grid of solutions / set of specifically distributed solutions [65], or a list of found peaks [67, 69].

Strength and Weakenesses

Since using detectors involves additional function evaluations, it might be required to identify an optimal number of detectors to maximize algorithm performance. A majority of existing methods just use one or a small number of detectors. However, in situations where only some parts of the search space change, e.g., see [73, 76, 82] and in a list of real-world problems cited in [70], using only a small number of detectors might not guarantee that changes are detected [82]. Recent attempts have been made to overcome this drawback. In [65, 70, 82], different methods were considered to study the optimal number of detectors depending on the size and complexity of the problem. A theoretical analysis in [65] showed that problem dimensionality is a prominent factor in the success of change detection. This finding was later confirmed by the experiments in [82].

One clear advantage of re-evaluting dedicated detectors is that it allows "robust detection" if a high enough number of detectors is used [82]. Richter [82] also showed that the more difficult the change detection is, the more favorable the approach of re-evaluating dedicated detectors is.

There are some disadvantages in re-evaluating dedicated detectors. First, there is the additional cost due to that detectors have to be re-evaluated at every generation. Second, this approach might not be accurate when used in problems with noisy fitness function because noises may mislead the algorithm to thinking that a change has occured [51]. It may also miss changes if changes did not occur in the region of detectors.

2.2.2.2 Detecting Changes Based on Algorithm Behaviors

Overview
Irregularities in algorithm behaviors can also be used to detect changes. In [31] (and many studies that follow the same idea), changes are detected based on monitoring the drop in the average of best found solutions over a number of generations. In a swarm-based study [51] where the swarm was divided into a tree-based hierarchy of sub-swarms, environmental changes were detected based on observation of changes in the hierarchy itself. In [65], the possibility of detecting changes based on diversity, and the relationship between the diversity of fitness values and the success rate of change detection, were studied. In [82], changes were detected based on statistical hypothesis tests to find the difference between distribution of the populations from two consecutive generations. This technique has been commonly used in environmental change detection in the real-world applications of biomedicine, data mining and image processing, as can be seen from the references cited in [82].

Strength and Weakenesses
The clear advantage of this approach is that it does not require any additional function evaluations. However, because no dedicated detector is used, there is no guarantee that changes are detected [82]. In addition, this approach may cause false positives and hence cause the algorithm to react unnecessarily when no change occurs. Evidence of false positives was found in [51, 76, 82]. Another possible disadvantage is that some methods following this approach might be algorithm-specific, such as the method of monitoring swarm hierarchy in [51].

2.2.3 Introducing Diversity When Changes Occur

2.2.3.1 Overview

In static environments, proper algorithm convergence is required so that the algorithm can focus on finding the best solution. In dynamic environments, however, convergence may result in negative effects. This is because if the dynamic landscape changes in one area and there is no member of the algorithm in this area, the change will become undetected. As a result, the algorithm will not be able to react to the change effectively and hence might fail to track the moving global optimum.

Intuitively one simple solution for this drawback is to increase the diversity of an EA after a change has been detected. This solution is described in the pseudo-code of Algorithm 1.

Algorithm 1 Introducing diversity after detecting a change

1. *Initialize:*: Initialize the population
2. *For each generation*
   a. *Evaluate*: Evaluate each member of the population
   b. *Check for changes*: Detect changes by monitoring possible signs of changes, e.g. a reduction in the best fitness values, or re-evaluation of old solutions
   c. *Increase diversity*: If change occurs, increase population's diversity by changing the mutations (sizes or rates) or relocating individuals
   d. *Reproduce*: Reproduce a new population using the adjusted mutation/learning/adaptation rate
   e. Return to step 2a

Diversity introduction can be done in many ways, for example, by increasing the current mutation rate as in hyper-mutation [31], by adding randomised individuals [40, 48], by increasing the mutation step size [107, 115], by moving individuals from one sub-population to another [40] or by keeping the sub-populations/individuals away from one another [50, 51].

Some of the first studies following this approach are hyper-mutation [31] and variable local search (VLS) [108, 109]. In his research, Cob [31] proposed an adaptive mutation operator called hyper-mutation whose mutation rate is a multiplication of the normal mutation rate and a hyper-mutation factor. The hyper-mutation is invoked only after a change is detected. In the VLS algorithm, the mutation step size is controlled by a variable local search range. This range is determined by the formula $(2^{BITS} - 1)$ where BITS is a value adjustable during the search [107] or adapted using a learning strategy borrowed from the feature partitioning algorithm by Vavak *et al.* [108].

In [70], hyper-mutation was used to solve dynamic constraint problems. Detectors are placed near the boundary of feasible regions and when the feasibility of these detectors changes, the EA increases its mutation rate to raise the diversity level to track the moving feasible regions. The mutation rate is decreased again once the moving feasible region has been tracked successfully. Riekert and Malan [86] proposed adaptive genetic programming which not only increases mutation, but also reduces elitism and increases crossover probability after a change. The idea of introducing diversity after a change has also been used in dynamic multi-objective optimization (DMO). For example, in a multi-population algorithm for DMO [40], when a change is detected, random individuals and some competitor individuals from other sub-populations are introduced to each sub-population to increase diversity.

Diversity introduction is also used in particle swarm optimization (PSO). Hu and Eberhart [48] introduced a simple mechanism in which a part of the swarm or the whole swarm will be re-diversified using randomization after a change is detected. Janson and Middendorf [51] followed a more sophisticated mechanism where in

addition to partial re-diversification, after each change the swarm is divided into several sub-swarms for a certain number of generations. The purpose of this is to prevent the swarm from converging to the old position of the global optimum too quickly. Daneshyari and Yen [34] proposed a cultural-based PSO where after a change, the swarms are re-diversified using a framework of knowledge inspired from the belief space in cultural algorithms. The diversity-introducing approach is still commonly used in many recent EDO algorithms, e.g., [67, 78, 82, 83, 85, 93, 115].

2.2.3.2 Strength and Weakness

In general, methods following the diversity-introducing approach appear to be good in solving problems with continuous changes where changes are small and medium. This is because invoking mutations or distributing individuals around an optimum resembles a type of "local search", which is useful to observe the nearby places of this optimum. Thus, if the optimum does not move very far, it might be tracked [107, 108].

However, this approach has some drawbacks that might make it not so suitable for certain type of problems. First, it is dependent on whether changes are known/easy to detect or not. If a change appears in a place where no individual exists, it will go undetected [65]. Second, it might be difficult to identify the right amount of diversity needed: too small steps will resemble local search while too large steps will result in random search [52]. Third, the approach might not be effective for solving problems with random changes or large changes (changes are severe) because many diversity-introducing methods have their mutation/relocation size restricted to a specific range. The diversity-introducing approach is still commonly used in many recent metaheuristics algorithms, e.g., [67, 70, 78, 82, 83, 85].

2.2.4 Maintaining Diversity during the Search

2.2.4.1 Overview

Another related approach is to maintain population diversity throughout the search process (see Algorithm 2). Methods following this approach do not detect changes

Algorithm 2 Maintaining diversity

1. *Initialize*: Initialize the population
2. *For each generation*
    a. *Evaluate*: Evaluate each member of the population
    b. *Maintain diversity*: Add a number of new, diversified individuals to the current population, or select more diversified individuals, or explicitly relocate individuals to keep them away from one another.
    c. *Reproduce*: Reproduce a new population
    d. Return to step 2a

explicitly. Instead, they rely on their diversity to adaptively cope with the changes. Diversity can be maintained by regularly introducing random individuals (*random immigrants*) to the population [43, 125], by sharing fitness [2], by specifically distributing some sentinel individuals [65], by explicitly keeping individuals from getting close to one another [10, 11], by dedicating one of the objectives for divesity purposes [24], or by combining several of the strategies above [1, 102].

In the random immigrants method [43], in every generation a number of generated random individuals are added to the population to maintain diversity. Experimental results show that the method is more effective in handling dynamics than regular EA [43]. It was also reported that the high diversity level brought by random immigrants also helps in handling constraints [70].

In [65], a different mechanism was proposed in which instead of generating random individuals, the sentinel placement method initializes a number of sentinels which are specifically distributed throughout the search space. Experiments show that this method might get better results than random immigrants and hypermutation in problems with large and chaotic changes [65].

Yang and Yao [125] proposed two other approaches based on the population-based incremental learning (PBIL) algorithm - parallel PBIL (PPBIL2) and dual PBIL (DPBIL). The PBIL algorithm has a probability vector adjusted based on the best found solutions. In PPBIL2, Yang and Yao [125] improved PBIL for DOPs by maintaining two parallel probability vectors: a vector similar to the original one in PBIL and a random initialized probability dedicated to maintain diversity during the search. To improve PBIL2 in dealing with large changes, Yang and Yao [125] proposed the DPBIL where two probability vectors are *dual* with each other, i.e., given the first vector P_1, the second vector P_2 is determined by $P_2[i] = 1 - P_1[i]$ $(i = 1, ..., n)$, where n is the number of variables. During the search only P_1 needs to learn from the best generated solution because P2 will change with P_1 automatically. PBIL and DPBIL were also combined with random-immigrants in [126] with better results than the original algorithms.

Another approach to maintain diversity is to reward individuals that are genetically different from their parents [110]. In this approach, in addition to a regular population, the algorithm maintains an additional population where individuals are selected based on their Hamming distance to their parents (to promote diversity) and another population where individuals are selected based on their fitness improvement compared to their parents (to promote exploitation). By observing its own performance in stagnation and population diversity, the algorithm adaptively adjusts the size of the three populations to react to dynamic environments.

Diversity can be maintained in evolution strategies (ESs) by preventing the strategy parameters from converging to zero, e.g. in [53]. The approach of maintaining diversity is also used in PSO to solve dynamic continuous problems. In their charged PSOs [9–11], Blackwell *et al.* applied a *repulsion* mechanism, which is inspired from the atom field, to prevent particles/swarms to get too close to each other. In this mechanism, each swarm is comprised of a nucleus and a cloud of charged particles which are responsible to maintain diversity. There is a repulsion among these particles to keep particles from approaching near to each other. In [34], both

the particle selection and replacement mechanisms are modified so that the most diversified particles (in term of Hamming distance) are selected and the particles that have similar positions are replaced. In the compound PSO [59], the degree of particles deviating from their original directions becomes larger when the velocities becomes smaller, and distance information was incorporated as one of the criteria to choose a particle for the update mechanism.

Bui *et al.* [24] used multiple objectives to maintain diversity. The dynamic problem is represented by a problem with two objectives: one original objective and one special objective created to maintain diversity. Similar examples can be found in [1, 102], of which the latter proposed six different types of objectives, including retaining more old solutions; retaining more random solutions; reversing the first objective; keeping a distance from the closest neighbour; keeping a distance from all individuals; and keeping a distance from the best individual. The diversity-maintaining strategy is still the main strategy in many recent approaches, for example, see [6, 9, 10, 29, 35, 39, 42, 50, 70, 125, 126].

2.2.4.2 Strength and Weakness

Methods following the diversity-maintaining approach may be good at solving problems with large changes (e.g. in [70, 73] random immigrants helped significantly improve the performance in dynamic constrained problems where changes are severe due to the presence of disconnected feasible regions), problems with slow changes (as shown in e.g. [2, 125]), and problems with competing peaks (as reported in [27]).

However, the diversity-maintaining approach might suffer from some disadvantages. First, continuously focusing on diversity may slow down, or even distract the optimization process [52]. Second, the approach may become less effective in dealing with small changes where the optima just take a slight move away from their previous places [32].

2.2.5 Memory Approaches

In situations where DOP changes are periodical or recurrent, and hence the optima may return to the regions near their previous locations, it might be useful to re-use previously found solutions to save computational time. The most common way to re-use old solutions in this manner is to maintain memory components in the algorithms. The memory can also play the role as a reserved place for storing old solutions in order to maintain diversity when needed. The memory can be integrated *implicitly* as a redundant representation in the algorithms, or maintained *explicitly* as a separate memory component.

2.2.5.1 Implicit Memory

The most common implicit memory used in EDO algorithms is redundant coding using diploid genomes, e.g., [41, 56, 68, 106, 121]. A diploid EA is usually an algorithm whose chromosomes contain two alleles at each locus. Although most

Algorithm 3 Multiploid EA for dynamic optimization

1. *Initialize*: Initialize the population and the multiploid representation
2. *For each generation*
    a. *Evaluate:* Evaluate each member of the population
    b. For each individual:
        i. *Detect changes*
        ii. *Adjust the dominance level of each allele* : If there is any change, adjust the dominance to accommodate the current change
        iii. *Select the dominant alleles according to their dominance level*
    c. *Reproduce*: Reproduce a new population using the adjusted mutations
    d. Return to step 2a

normal EAs for static problems are haploid, it is believed that diploid, and other multiploid approaches, are suitable for solving dynamic problems [56]. A pseudo code for multiploid approaches in dynamic environments is described in Algorithm 3, where the following three components need to be incorporated: (i) represent the redundant code; (ii)readjust the dominance of alleles; and (iii) check for changes.

The dominance of alleles is usually represented by a table [68, 91] or a mask [33] mapping between genotypes and phenotypes. The dominance then can be changed adaptively among alleles depending on the detection of changes in the landscape.

2.2.5.2 Explicit Memory

Methods that use explicit memory generally follow the steps in Algorithm 4. The memory can be maintained *directly* in the form of previous good solutions [8, 15, 34, 55, 60, 62, 64, 93, 118, 119, 126–128], or it can be maintained *indirectly* in the form of associative information. Various type of associative information can be included, e.g., the environment at the considered time [37, 79]; the list of environmental states

Algorithm 4 Using explicit memory

1. *Initialize*:
    a. Initialize the population
    b. Initialize the explicit memory
2. *For each generation*
    a. Evaluate each member of the population
    b. Update the memory
    c. Reproduce a new population
    d. Use information from the memory to update the new population
    e. Return to step 2a

and state transition probabilities [96]; the successful individuals for certain types of changes[94, 120]; the probability vector that created the best solutions [126]; the distribution statistics information of the population at the considered time [119]; the probability of the occurrence of good solutions in each area of the landscape [84, 85]; or the probability of likely feasible regions [83].

Generally the best found elements (direct or associative) of the current generation will be used to update the memory. These newly found elements will replace some existing elements in the memory, which can be the oldest member [37, 95, 103, 115], the one with the least contributions to the diversity of the population [15, 37, 95, 118, 126], or the one with least contribution to fitness [37]. During the search, usually the best elements in the memory (i.e. the ones that show the best results when being re-evaluated) will be used to replace the worst individuals in the population. Replacement can take place after each generation or after a certain number of generations, or it can be done after each change if the change can be detected.

2.2.5.3 Strength and Weakness

Memory methods are particularly effective for solving problems with periodically changing environments. For example, it was shown that the memory-based versions of EAs and random-immigrant significantly outperform the original algorithms in cyclic dynamic environments [122]. The approach might also be good in slowing down convergence and favour diversity [16, 18].

Memory methods, however, have disadvantages that may require them to be used with some other methods for the best results. First, they might be useful only when optima reappear at their previous locations or if the environment returns to its previous states [15, 16]. Second, they might not be good enough to maintain diversity for the population [15]. Third, the information stored in the memory might become redundant (and obsolete) and consequently may affect the performance of the algorithm. In addition, redundant coding approaches might not be good for cases where the number of oscillating states is large.

2.2.6 Prediction Approaches

2.2.6.1 Overview

In DOPs where changes exhibit regular patterns, it might be helpful to try to learn the patterns from previous search experience, and then try to predict changes in the future. A pseudo code of methods following this approach is shown in Algorithm 5.

One of the common prediction approaches is to predict the movement of the moving optima. Hatzakis and Wallace [45] combined a forecasting technique (autoregressive) with an EA to predict the location of the next optimal solution after a change is detected. The forecasting model (time series model) is created using a sequence of optimum positions found in the past. Experimental results show that if this algorithm can predict the movements of optima correctly, it can work well with

Algorithm 5 Prediction approach to solve dynamic problems

1. *Initialize phase*:
    a. Initialize the population
    b. Initialize the learning model and training set
2. *Search for optimum solutions and detect changes*
3. *If a change is detected*
    a. Use the current environment state as the input for the learning model
    b. Use the learning model to estimate the type of this current change and/or how the next change should be
    c. Generate new individuals or recall old ones that best match with the estimation
    d. Search for the new optimum using the new population
    e. Update the training set based on the search results
4. Return to step 2

very fast changes. A similar approach was proposed in [90] where the movement of optima was predicted using Kalman filters. The predicted information (the next location of the optimum) is incorporated into an EA in three ways: First, the mutation operator is modified by introducing some bias so that individuals' exploration is directed toward the predicted region. Second, the fitness function is modified so that individuals close to the estimated future position are rewarded. Third, some "gifted" individuals are generated at the predicted position, and introduced into the population to guide the search. Experiments on a visual tracking benchmark problem show that the proposed method does improve the tracking of the optimum, both in terms of distance to the real optimum and smoothness of the tracking.

Prediction was also used to determine the locations that individuals should be re-initialized to when a change occurs. In [129] this approach is used to solve two dynamic multi-objective optimization benchmark problems in two ways: First, the solutions in the Pareto set from the previous change periods were used as a time series to predict the next re-initialization locations. Second, to improve the chance of the initial population to cover the new Pareto set, the predicted re-initialization population is perturbed with a Gaussian noise whose variance is estimated based on historical data. Compared with random-initialization, the approach was able to achieve better results on the two tested problems.

Another interesting approach is to predict the time when the next change will occur and which possible environments will appear in the next change [96, 97]. In these works, the authors used two prediction modules to predict two different factors. The first module, which uses either a linear regression [96] or a non-linear regression [97], is used to estimate the generation when the next change will occur. The second module, which uses Markov chain, monitors the transitions of previous environments and based on this data provides estimations of which environment will

appear in the next change. Experimental results show that an EA with the proposed predictor is able to perform better than a regular EA in cyclic/periodic environments.

A special class of prediction approaches is dynamic time-linkage optimization [12–14, 72, 74]. Time-linkage problems are problems where the current solutions made by the algorithms can influence the future dynamics. In such problems, it was suggested that the only way to solve the problems effectively is to predict future changes and take into account the possible future outcomes when solving the problems online. Research in [12–14, 72, 74] followed this idea to solve time-linkage problems effectively. Another related study is the anticipation approach [20] in solving dynamic scheduling problems where in addition to finding good solutions, the solver also tries to move the system "into a flexible state" where adaptation to changes can be done more easily. Specifically, because it is observed that in the tested dynamic job-shop scheduling problem, the flexibility of the system can be increased by avoiding early machine idle times, the authors proposed a scheduling approach where in addition to the main optimality objective, solutions with early idle time are penalized. The experimental results show that such an anticipation approach significantly improved the performance of the system.

2.2.6.2 Strength and Weakness

Prediction approaches may become very effective if their predictions are accurate. In this case, the algorithms can detect/track/find the global optima quickly, as shown in [45, 94, 120]. However, prediction/adaptation-based algorithms also have their own disadvantages, mostly due to training errors. These errors might be resulted by the unpredictable nature of the problems. If the changes are stochastic, or history data are misleading, prediction approaches might not get satisfactory results. For example, [72, 74] illustrated a situation where history data are actually inappropriate for the prediction and might even mislead the predictor to get worse results.

Prediction errors might also be due to wrong training data, or lack of training data. As in the case of any learning/predicting/forecasting model, the algorithms may need a large enough set of training data to produce good results. It also means that the prediction can only be started after sufficient training data have been collected, e.g., [12, 13, 96, 97]. In the case of dynamic optimization where there is a need of finding/tracking the optima as quick as possible, this might be a disadvantage.

2.2.7 Self-adaptive Approaches

In certain cases, the self-adaptive mechanisms of EAs and other meta-heuristics can be used effectively to cope with changes. One example is the GA with genetic mutation rate [44], which allows the algorithm to evolve its own mutation strategy parameters during the search process based on the fitness of the population. In this method, the mutation rate is encoded in genes and is influenced by the selection process. The algorithm was tested in both gradual and abrupt dynamic landscapes. The results show that the algorithm has better performance than a conventional GA.

However, it is still not better than hyper-mutation (see section 2.2.3 and [31]). A similar method was proposed by Ursem in his multinational GA (MGA) [104]. Five different parameters (probability for mutation, probability for crossover, selection ratio, mutation variance and distance) are encoded in the genomes of his MGA for adaptation. The adaptation mechanism works well in simple cases where the velocity of moving peaks is constant. However, in cases where the velocity is not constant, the adaptation seems to be not fast enough. These two results show the difficulty of applying adaptive parameter tuning to complex dynamic optimization.

The self-adaptive mechanisms of such EAs as ES or evolutionary programming (EP) were also investigated for using in dynamic optimization. Angeline [3] examined self-adaptive EP (saEP) and showed that the strategy is not effective for all types of tested problems. Bäck [7] showed that the log-normal self-adaptation in ES may perform better than saEP. Experiments pointed out that algorithm implementation and parameter settings have much less influence on ES in dynamic environments than in stationary environments [92] and that ES might be unreliable in rapidly changing environments [114]. Weicker [112] also argued that it is possible that if Gaussian mutation is used in the standard ES, self-adaptation might not be appropriate for dynamic optimization.

Some mathematical analyses on the performance of self-adaptive ES in dynamic environments were proposed. Arnold and Beyer [4] pointed out that the cumulative mutation step-size adaptation of ES can work well on a variant of the sphere model with random dynamics of the optimum. The strategy can realize the optimal mutation step-size for the model. However, in the sphere model with linear dynamics, another research of Arnold and Beyer [5] revealed that the mutation step-size realized by ES is not the optimal one (but the adaptation still ensures that the optimum can be tracked).

2.2.8 Multi-population Approaches

2.2.8.1 Overview

Multi-population approach, which maintains multiple sub-populations concurrently, can be seen as a combination of diversity maintaining/introducing, memory and adaptation. Each sub-population may handle a separate area of the search space. Each of them may also take responsibility for a separate task. For example, some sub-populations may focus on searching for the global optimum while some others may concentrate on tracking any possible changes. These two types of populations then may communicate with each other to bias the search. A typical pseudo-code of the multi-population approach is shown in Algorithm 6.

Methods following the approach of using multiple populations usually need to accomplish two goals: First, they may need to assign different types of tasks to different sub-populations, for example, P_{search} to search and P_{track} to track, so that the search can be done effectively. Second, they need to divide the sub-populations appropriately and make sure that the sub-populations are not overlapped to have the

Algorithm 6 Multi-population approach

1. *Initialize*:
    a. Initialize the set P_{search} of sub populations finding the global optima
    b. Initialize the set P_{track} of sub populations tracking changes in the landscape
2. *For each generation*:
    a. *Search for optima*: Sub-populations in P_{search} find the global optima
    b. *Track changes*: Sub-populations in P_{track} track any changes
    c. *Maintain diversity*: Re-allocate/split/merge the sub-populations so that they are not overlapped and can cover a larger area of the search space
    d. *Adjust*: Re-adjust each sub-population in P_{search} based on the experience from sub-populations in P_{track}
    e. Reproduce each sub-population
    f. Return to step 2a

best diversity and also to avoid the situation where many sub-populations find the same peak.

For the first goal, *assigning different tasks to the sub-populations*, there might be multiple small populations in P_{search} searching for new solutions and there is only one large population in P_{track} to track changing peaks [77], or there might be one large population to search and multiple sub-population for tracking changes [9, 19, 29, 38, 61, 63, 73], or each sub-population can both search for new solutions and track changes [39, 57, 58, 104]. Relating to the goal of assigning the tasks to sub-populations, it should be noted that in dynamic optimization multiple populations are used not only for exploring different parts of the search space, but also for co-evolution [40, 73, 75] or maintaining diversity and balancing between exploitation and exploration [110].

For the second goal, *dividing the sub-populations and making sure that the sub-populations are not overlapping*, there are different approaches, of which the most common is clustering: choosing some solutions in the population as the centres of the future clusters, then defining each sub-population as a hyper-cube or sphere with a given size. All individuals within the range of a hyper-cube/sphere will belong to the corresponding sub-population of that hyper-cube/sphere. For example, the self-organizing scouts (SOS) algorithm [19] keeps the sub-populations from being overlapped by confining each sub-population to a hyper-cube determined by a centre (the most fit individual in the population) and a pre-defined range. If an individual of one sub-population ventures to the area monitored by another sub-population, this individual will simply be discarded and re-initialized (this process is called *exclusion*). The same approach is also used in DE [61, 63] and PSO [10, 58]. For example, in multi-swarm PSO (mPSO) [10], swarms are divided into sub-swarms so that each swarm watches a different peak. In addition, mPSO also maintains a similar mechanism (named anti-convergence) to the P_{search} in SOS so that there is always one free swarm to continue exploring the search space. Another example is

the speciation PSO (SPSO) algorithm [58], where each species is a hyper-sphere whose centre is the best-fit individual in the species and each species can be used to track a peak. In recent clustering approaches [57, 115], density-based clustering methods are also used to divide/separate the sub-populations and to allow the algorithms explore different parts of the search landscape.

Other approaches to divide sub-populations are to incorporate some mechanism of penalty/rewarding to keep the sub-populations apart [77], and to estimate the basins of attractions of peaks and use these basins as separate regions for each sub-population [104].

2.2.8.2 Strength and Weakness

Multi-population approaches are thought to have multiple advantages. First, they can maintain enough diversity to adaptively start a new search whenever a new change appears, as illustrated in [17]. Second, they may be able to recall some information from the previous generations thanks to one (or several) population(s) dedicated for retaining old solutions, as shown in their good performance in problems with recurrent changes [15, 104]. Third, they can search/track the moves of multiple optima, as analysed in many existing studies on multi-populations, e.g., [17] and [104]. Finally, they can be very effective for solving problems with competing peaks or multimodal problems. A survey by Moser [66] showed that among 19 surveyed algorithms that are designed to solve the moving peaks benchmark (MPB) [15] with multimodal competing peaks, a majority (15 out of 19) follow the multi-population approach.

There are also disadvantages in using multi-population approaches. First, too many sub-populations may slow down the search. For example, Blackwell and Branke [10] showed that for their multi-swarm PSO algorithm, if the number of sub-populations (swarms) is larger than the number of peaks, the performance of the algorithm decreases. It might also be difficult to identify the appropriate number of sub-populations, as well as the size of each sub-population. Second, the need of calculating the distance/similarity/regional metrics to separate the sub-populations might also affect the performance. Third, in academic research, multi-population approaches have been tested mostly in the continuous domain, and hence more evidence might be needed to confirm their effectiveness on combinatorial problems.

2.3 Theoretical Development of EDO Methodologies

EDO research so far has mainly been empirical. Most theoretical analysis of EDO has just started in recent years with some results. Analysing EAs for DOPs is considered more difficult than analysing EAs for static problems due to the extra dynamics introduced in DOPs. The theoretical studies on EDO methodologies are briefly reviewed as follows.

Initial EDO theoretical works were extensions of the analysis of simple EAs, e.g., the (1+1) EA[3], for static optimization to simple DOPs, e.g., the dynamic bit matching problem [99]. In [99], the authors presented the transition probabilities of the (1+1) EA and showed that even small perturbations in the fitness function could have a significantly negative impact on the performance of the (1+1) EA. Based on this work, Branke and Wang [23] developed an analytical model for a (1, 2)-ES and compared different strategies to handle an environmental change within a generation on the dynamic bit matching problem .

The first hitting time of a (1+1)-ES was analyzed by Droste [36] on the dynamic bit matching problem, where exactly one bit is changed with a given probability p after each function evaluation. It was shown that the expected first hitting time of the (1+1)-ES is polynomial if and only if $p = O(logn/n)$. Arnold and Beyer [4] investigated the tracking behaviour of an $(\mu/\mu, \lambda)$-ES with self-adaptive mutation step-size on a single continuously moving peak. They derived a formula to predict the tracking distance of the population from the target. Jansen and Schellbach [49] presented a rigorous performance analysis of the $(1+\lambda)$-EA on a tracking problem in a two-dimensional lattice and showed that the expected first hitting time strictly increases with the offspring population size (i.e., λ) whereas the expected number of generations to reach the target decreases with λ. In [114], Weicker and Weicker analyzed the behaviour of ESs with several mutation variants on a simple rotating dynamic problem. In [111], Weicker presented a framework for classifying DOPs and used it to analyze how the offspring population size and two special techniques for DOPs affect the tracking probability of a $(1,\lambda)$-ES. Weicker [113] also used Markov models to analyze the tracking behaviour of $(1,\lambda)$-ESs with different mutation operators for a discrete optimization problem with a single moving optimum.

Rohlfshagen *et al.* [87] analyzed how the magnitude and frequency of changes may affect the performance of the (1+1)-EA on two specially designed pseudo-Boolean functions under the dynamic framework of the XOR DOP generator [118]. They demonstrated two counter-intuitive results, i.e., the algorithm is efficient if the magnitude of change is large and inefficient when the magnitude of change is small, and the algorithm is efficient if the frequency of change is very high and inefficient if the frequency of change is sufficiently low. These results allow us to gain a better understanding of how the dynamics of a function may affect the runtime of an algorithm.

In addition to the above runtime analysis of EDO methodologies, there are also theoretical analysis of dynamic fitness landscape [21, 22, 80, 81, 83, 84, 88, 89, 100, 101]. Readers are referred to [71] for a literature review on research in this area.

[3] In a (1+1) EA, there is only one solution maintained in the population. In each iteration, the unique solution acts as the parent to generate an offspring via mutation. If the fitness of the offspring is not worse than the parent, the offspring will replace the parent; otherwise, the parent will survive into the next generation.

2.4 Summary and Future Research Directions

2.4.1 Summary

The review above showed that each EDO approach seems to be suitable only for certain types of DOPs, which conforms to the No Free Lunch theorem [116]. The fact that each approach is likely to be suitable to some particular classes of problems is also the reason why many recent studies try to combine different approaches into one single algorithm to solve the problems better. Overall, multi-population approaches seem to be the most flexible approach to date in the continuous domain.

The review showed that there have been some recent works on the theory behind EDO. These theoretical studies are still quite basic. However, they have made very important first steps toward understanding EDO and will surely act as the basis for further theoretical studies on EDO.

It should be noted that most existing EDO methods were tested and evaluated on academic problems only. This leads to the question of whether these methods would still be effective in real-world DOPs. In the next subsection, we will discuss this question in detail.

2.4.2 The Gaps between Academic Research and Real-World Problems

The lack of a clear link between EDO academic research and real-world scenarios has lead to some criticisms on how realistic current academic problems are. Ursem *et al.* [105] questioned the importance of current academic benchmarks by stating that "no research has been conducted to thoroughly evaluate how well they reflect characteristic dynamics of real-world problems"; Branke *et al.* [22] pointed out that "little has been done to characterize and understand the nature of a change in real-world problems"; Rohlfshagen and Yao [88] criticized that "a large amount of effort is directed at an academic problem that may only have little relevance in the real world"; and in [74, 76], it has been showed that there are some classes of real-world problems whose characteristics have not been captured by existing academic research yet. Nguyen and Yao [76] also showed evidence of situations where existing EDO techniques could not solve certain classes of DOPs effectively due to the uncaptured characteristics of DOPs.

Recently, a detailed analysis [70, Chapter 3] of a large set of recent "real"[4] real-world DOPs has been made to investigate the characteristics of real-world problems and how they relate to the characteristics of current academic benchmark problems. This investigation pointed out certain gaps between academic EDO research and real-world DOPs. First, current studies in academic EDO do not cover all types of com-

[4] Only references that actually use real-world data or solve problems in actual real-world situations were considered. Benchmark problems, even if designed to simulate real-world applications, were not considered unless there is evidence that the data used to create the benchmark were taken from real-world applications.

mon DOPs yet. There are two types of problems that are very common in real-world situations but received very little attention from the community: dynamic constrained problems and time-linkage problems. Second, although many current EDO academic research works only focus on one major optimization goal: optimality (to find the best fitness value), the study in [70] showed that there might be many other common optimization goals. Third, although most current EDO benchmark problems have only one changing factor, the study in [70] showed that there are also other common types of changing factors: constraints, number of variables, domain ranges, etc.

In summary, the review in [70] showed that besides the characteristics and assumptions commonly used in EDO academic research, real-world DOPs also have other important types of problems and problem characteristics that have not been studied extensively by the EDO community. In order to solve real-world DOPs more effectively, it is necessary to take these characteristics and problem types into account when designing new methodolgoies.

2.4.3 Future Research Directions

As reviewed in this chapter, there have been many studies devoted to EDO methodolgoies and fruitful results have been achieved. However, the research domain of EDO is still relatively young. Much more effort is needed to fully develop and understand the domain of EDO. Some future research directions on EDO methodologies are highlighted and suggested as follows.

First, although a number of EDO approaches have been developed for solving DOPs, new efficient approaches are of great needs. As the review has shown, different methods have different strength and weakness on different DOPs. Hence, it is also worthy to further develop and investigate hybrid methods for DOPs in the future. Here, it is very important to develop adaptive systems that can deal with DOPs of different characteristics. Active adaptability should also be addressed so that future algorithms are able to effectively handle dynamics even without change detection.

Second, as shown earlier in this chapter, most EDO methodologies focus on solving academic problems, where there is no clear link to real-world characteristics. Although there have been some real-world application studies on EDO, e.g., see [28–30, 70, 117] (also see [70, 123] for detailed lists of references), the number of EDO application studies so far is still very limited. One of the important research direction for the EDO community is to consider and model more real-world DOPs, and apply EDO and other meta-heuristic methods to solve them in the future. This will further enhance the applicability and feasibility of EDO in practical situations.

Third, as discussed earlier, theoretical research on EDO is still at the beginning stage. The relative lack of theoretical proof on EDO makes it difficult to evaluate the strength and weakness of EDO algorithms on solving different types of DOPs. As reviewed, the computational complexity analysis of EDO has started with some promising results. However, this area of study needs to be extended significantly to gain more insight as to which DOP is difficult or easy to solve for what types of EDO methods. Here, techniques for analyzing evolutionary optimization on static

problems, e.g., drift analysis [46, 47], may be applied or adapted to analyze EDO. It will also be beneficial to analyse the dynamic behavour of EDO algorithms and how the behavour satisfies the real-world practitioners' requirements, which are not always the ability to identify/track the global optimum after each change.

Acknowledgements. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant numbers EP/E058884/1, EP/K001523/1, EP/E060722/1, and EP/K001310/1, a UK ORS Award, a studentship from the School of Computer Science, University of Birmingham, and an EU-funded project named "Intelligent Transportation for Dynamic Environment (InTraDE)".

References

[1] Abbass, H.A., Deb, K.: Searching under multi-evolutionary pressures. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 391–404. Springer, Heidelberg (2003)

[2] Andersen, H.C.: An investigation into genetic algorithms, and the relationship between speciation and the tracking of optima in dynamic functions. Honours thesis, Queensland University of Technology, Brisbane, Australia (1991)

[3] Angeline, P.J.: Tracking extrema in dynamic environments. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 335–345. Springer, Heidelberg (1997)

[4] Arnold, D.V., Beyer, H.-G.: Random Dynamics Optimum Tracking with Evolution Strategies. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 3–12. Springer, Heidelberg (2002)

[5] Arnold, D.V., Beyer, H.G.: Optimum tracking with evolution strategies. Evol. Comput. 14(3), 291–308 (2006)

[6] Azevedo, C., Araujo, A.: Generalized immigration schemes for dynamic evolutionary multiobjective optimization. In: Proc. 2011 IEEE Congr. Evol. Comput., pp. 2033–2040 (2011)

[7] Bäck, T.: On the behavior of evolutionary algorithms in dynamic environments. In: Proc. 1998 IEEE Int. Conf. on Evol. Comput., pp. 446–451 (1998)

[8] Bendtsen, C.N., Krink, T.: Dynamic memory model for non-stationary optimization. In: Proc. 2002 IEEE Congr. Evol. Comput., pp. 145–150 (2002)

[9] Blackwell, T.: Particle swarm optimization in dynamic environment. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 28–49. Springer, Heidelberg (2007)

[10] Blackwell, T., Branke, J.: Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans. Evol. Comput. 10(4), 459–472 (2006)

[11] Blackwell, T.M., Bentley, P.J.: Dynamic search with charged swarms. In: Proc. 2002 Genetic and Evol. Comput. Conf., pp. 19–26 (2002)

[12] Bosman, P.A.N.: Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Yang, S., Branke, J. (eds.) GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization (2005)

[13] Bosman, P.A.N.: Learning and anticipation in online dynamic optimization. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 129–152. Springer, Heidelberg (2007)
[14] Bosman, P.A.N., Poutré, H.L.: Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In: Proc. 2002 Genetic and Evol. Comput. Conf., pp. 1165–1172 (2007)
[15] Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proc. 1999 IEEE Congr. Evol. Comput., vol. 3, pp. 1875–1882 (1999)
[16] Branke, J.: Evolutionary approaches to dynamic environments - updated survey. In: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 27–30 (2001)
[17] Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer (2001)
[18] Branke, J.: Evolutionary approaches to dynamic optimization problems – introduction and recent trends. In: Branke, J. (ed.) GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 2–4 (2003)
[19] Branke, J., Kaußler, T., Schmidth, C., Schmeck, H.: A multi-population approach to dynamic optimization problems. In: Proc. 4th Int. Conf. Adaptive Comput. Des. Manuf., pp. 299–308 (2000)
[20] Branke, J., Mattfeld, D.: Anticipation and flexibility in dynamic scheduling. Int. J. of Production Research 43(15), 3103–3129 (2005)
[21] Branke, J., Orbayı, M., Uyar, Ş.: The role of representations in dynamic knapsack problems. In: Rothlauf, F., et al. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 764–775. Springer, Heidelberg (2006)
[22] Branke, J., Salihoglu, E., Uyar, Ş.: Towards an analysis of dynamic environments. In: Proc. 2005 Genetic and Evol. Comput. Conf., pp. 1433–1439 (2005)
[23] Branke, J., Wang, W.: Theoretical analysis of simple evolution strategies in quickly changing environments. In: Proc. 2003 Genetic and Evol. Comput. Conf., pp. 537–548 (2003)
[24] Bui, L., Abbass, H., Branke, J.: Multiobjective optimization for dynamic environments. In: Proc. 2005 IEEE Congr. Evol. Comput., vol. 3, pp. 2349–2356 (2005)
[25] Carlisle, A., Dozier, G.: Adapting particle swarm optimisationto dynamic environments. In: Proc. 2000 Int. Conf. on Artif. Intell., pp. 429–434 (2000)
[26] Carlisle, A., Dozier, G.: Tracking changing extrema with adaptive particle swarm optimizer. In: Proc. 5th World Automation Congr., vol. 13, pp. 265–270 (2002)
[27] Cedeno, W., Vemuri, V.R.: On the use of niching for dynamic landscapes. In: Proc. 1997 IEEE Int. Conf. on Evol. Comput. (1997)
[28] Cheng, H., Yang, S.: Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks. Eng. Appl. of Artif. Intell. 23(5), 806–819 (2010)
[29] Cheng, H., Yang, S.: Multi-population genetic algorithms with immigrants scheme for dynamic shortest path routing problems in mobile ad hoc networks. In: Di Chio, C., et al. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 562–571. Springer, Heidelberg (2010)
[30] Chitty, D.M., Hernandez, M.L.: A hybrid ant colony optimisation technique for dynamic vehicle routing. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 48–59. Springer, Heidelberg (2004)
[31] Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuouis, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA (1990)

[32] Cobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: Proc. 1993 Int. Conf. on Genetic Algorithms, pp. 523–530 (1993)
[33] Collingwood, E., Corne, D., Ross, P.: Useful diversity via multiploidy. In: Proc. 1996 IEEE Int. Conf. on Evol. Comput., pp. 810–813 (1996)
[34] Daneshyari, M., Yen, G.: Dynamic optimization using cultural based pso. In: Proc. 2011 IEEE Congr. Evol. Comput., pp. 509–516 (2011)
[35] Deb, K., Rao N., U.B., Karthik, S.: Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 803–817. Springer, Heidelberg (2007)
[36] Droste, S.: Analysis of the (1+1) ea for a dynamically changing onemax-variant. In: Proc. 2002 IEEE Congr. Evol. Comput., pp. 55–60 (2002)
[37] Eggermont, J., Lenaerts, T., Poyhonen, S., Termier, A.: Raising the dead: Extending evolutionary algorithms with a case-based memory. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tetamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 280–290. Springer, Heidelberg (2001)
[38] Fernández, J.L., Arcos, J.L.: Adapting particle swarm optimization in dynamic and noisy environments. In: Proc. 2010 IEEE Congr. Evol. Comput., pp. 765–772 (2010)
[39] de França, F.O., Von Zuben, F.J.: A dynamic artificial immune algorithm applied to challenging benchmarking problems. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 423–430 (2009)
[40] Goh, C.K., Tan, K.C.: A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans. on Evol. Comput. 13(1), 103–127 (2009)
[41] Goldberg, D.E., Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Proc. Int. Conf. on Genetic Algorithms, pp. 59–68 (1987)
[42] Gouvêa Jr., M., Araújo, A.: Adaptive evolutionary algorithm based on population dynamics for dynamic environments. In: Proc. 2011 Genetic and Evol. Comput. Conf., pp. 909–916 (2011)
[43] Grefenstette, J.J.: Genetic algorithms for changing environments. In: Proc. 2nd Int. Conf. Parallel Problem Solving from Nature, pp. 137–144 (1992)
[44] Grefenstette, J.J.: Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In: Proc. 1999 IEEE Congr. Evol. Comput., vol. 3, pp. 2031–2038 (1999)
[45] Hatzakis, I., Wallace, D.: Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In: Proc. 2006 Genetic and Evol. Comput. Conf., pp. 1201–1208 (2006)
[46] He, J., Yao, X.: From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. IEEE Trans. Evol. Comput. 6(5), 495–511 (2002)
[47] He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3(1), 21–35 (2004)
[48] Hu, X., Eberhart, R.: Adaptive particle swarm optimisation: detection and response to dynamic systems. In: Proc. 2002 IEEE Congr. Evol. Comput., pp. 1666–1670 (2002)
[49] Jansen, T., Schellbach, U.: Theoretical analysis of a mutation-based evolutionary algorithm for a tracking problem in lattice. In: Proc. 2005 Genetic and Evol. Comput. Conf., pp. 841–848 (2005)
[50] Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer and its adaptive variant. IEEE Trans. Syst., Man, and Cybern.-Part B: Cybern. 35, 1272–1282 (2005)

[51] Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer for noisy and dynamic environments. Genetic Programming and Evolvable Machines 7(4), 329–354 (2006)
[52] Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments—a survey. IEEE Trans. Evol. Comput. 9(3), 303–317 (2005)
[53] Jin, Y., Sendhoff, B.: Constructing dynamic optimization test problems using the multi-objective optimization concept. In: Raidl, G.R., et al. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 525–536. Springer, Heidelberg (2004)
[54] Kramer, G.R., Gallagher, J.C.: Improvements to the *CGA enabling online intrinsic in compact EH devices. In: Proc. 2003 NASA DoD Conf. on Evolvable Hardware, pp. 235–231 (2003)
[55] Lepagnot, J., Nakib, A., Oulhadj, H., Siarry, P.: Brain cine mri segmentation based on a multiagent algorithm for dynamic continuous optimization. In: Proc. 2011 IEEE Congr. Evol. Comput., pp. 1695–1702 (2011)
[56] Lewis, J., Hart, E., Ritchie, G.: A comparison of dominance mechanisms and simple mutation on non-stationary problems. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 139–148. Springer, Heidelberg (1998)
[57] Li, C., Yang, S.: A clustering particle swarm optimizer for dynamic optimization. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 439–446 (2009)
[58] Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: Proc. 2006 Genetic and Evol. Comput. Conf., pp. 51–58 (2006)
[59] Liu, L., Wang, D., Yang, S.: Compound particle swarm optimization in dynamic environments. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 616–625. Springer, Heidelberg (2008)
[60] Louis, S.J., Xu, Z.: Genetic algorithms for open shop scheduling and re-scheduling. In: Cohen, M.E., Hudson, D.L. (eds.) Proc. ISCA 11th Int. Conf. on Computers and their Applications, pp. 99–102 (1996)
[61] Lung, R.I., Dumitrescu, D.: A new collaborative evolutionary-swarm optimization technique. In: Proc. 2007 Genetic and Evol. Comput. Conf., pp. 2817–2820 (2007)
[62] Mavrovouniotis, M., Yang, S.: Memory-based immigrants for ant colony optimization in changing environments. In: Di Chio, C., et al. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 324–333. Springer, Heidelberg (2011)
[63] Mendes, R., Mohais, A.: Dynde: a differential evolution for dynamic optimization problems. In: Proc. 2005 IEEE Congr. Evol. Comput., pp. 2808–2815 (2005)
[64] Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 149–158. Springer, Heidelberg (1998)
[65] Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Springer, Berlin (2004) ISBN 3-540-21231-0
[66] Moser, I.: Review - all currently known publications on approaches which solve the moving peaks problem. Tech. Rep., Swinburne University of Technology, Melbourne, Australia (2007)
[67] Moser, I., Hendtlass, T.: A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. In: Proc. 2007 IEEE Congr. Evol. Comput., pp. 252–259 (2007)

[68] Ng, K.P., Wong, K.C.: A new diploid scheme and dominance change mechanism for non-stationary function optimization. In: Proc. 6th Int. Conf. on Genetic Algorithms, pp. 159–166 (1995)
[69] Nguyen, T.T.: Tracking optima in dynamic environments using evolutionary algorithms - rsmg report 5. Tech. Rep., School of Computer Science, University of Birmingham (2008), http://www.cs.bham.ac.uk/~txn/unpublished/reports/Report_5_Thanh.pdf
[70] Nguyen, T.T.: Continuous Dynamic Optimisation Using Evolutionary Algorithms. Ph.D. thesis, School of Computer Science, University of Birmingham (2011), http://etheses.bham.ac.uk/1296 and http://www.staff.ljmu.ac.uk/enrtngu1/theses/phd_thesis_nguyen.pdf
[71] Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evol. Comput. 6, 1–24 (2012)
[72] Nguyen, T.T., Yang, Z., Bonsall, S.: Dynamic time-linkage problems - the challenges. In: IEEE RIVF Int. Conf. on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, pp. 1–6 (2012)
[73] Nguyen, T.T., Yao, X.: Benchmarking and solving dynamic constrained problems. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 690–697 (2009)
[74] Nguyen, T.T., Yao, X.: Dynamic time-linkage problems revisited. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 735–744. Springer, Heidelberg (2009)
[75] Nguyen, T.T., Yao, X.: Solving dynamic constrained optimisation problems using stochastic ranking and repair methods. IEEE Trans. Evol. Comput. (2010) (submitted), http://www.staff.ljmu.ac.uk/enrtngu1/Papers/Nguyen_Yao_dRepairGA.pdf
[76] Nguyen, T.T., Yao, X.: Continuous dynamic constrained optimisation - the challenges. IEEE Trans. Evol. Comput. 16(6), 769–786 (2012)
[77] Oppacher, F., Wineberg, M.: The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment. In: Proc. 1999 Genetic and Evol. Comput. Conf., vol. 1, pp. 504–510 (1999)
[78] Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Comput. 10(4), 440–458 (2006)
[79] Ramsey, C.L., Grefenstette, J.J.: Case-based initialization of genetic algorithms. In: Proc. 5th Int. Conf. on Genetic Algorithms, pp. 84–91 (1993)
[80] Richter, H.: Behavior of evolutionary algorithms in chaotically changing fitness landscapes. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 111–120. Springer, Heidelberg (2004)
[81] Richter, H.: Evolutionary optimization in spatio–temporal fitness landscapes. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 1–10. Springer, Heidelberg (2006)
[82] Richter, H.: Detecting change in dynamic fitness landscapes. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 1613–1620 (2009)
[83] Richter, H.: Memory design for constrained dynamic optimization problems. In: Di Chio, C., et al. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 552–561. Springer, Heidelberg (2010)
[84] Richter, H., Yang, S.: Memory based on abstraction for dynamic fitness functions. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 596–605. Springer, Heidelberg (2008)

[85] Richter, H., Yang, S.: Learning behavior in abstract memory schemes for dynamic optimization problems. Soft Comput. 13(12), 1163–1173 (2009)
[86] Riekert, M., Malan, K.M., Engelbrecht, A.P.: Adaptive genetic programming for dynamic classification problems. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 674–681 (2009)
[87] Rohlfshagen, P., Lehre, P.K., Yao, X.: Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change. In: Proc. 2009 Genetic and Evol. Comput. Conf., pp. 1713–1720 (2009)
[88] Rohlfshagen, P., Yao, X.: Attributes of dynamic combinatorial optimisation. In: Li, X., et al. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 442–451. Springer, Heidelberg (2008)
[89] Rohlfshagen, P., Yao, X.: On the role of modularity in evolutionary dynamic optimisation. In: Proc. 2010 IEEE Congr. Evol. Comput., pp. 3539–3546 (2010)
[90] Rossi, C., Abderrahim, M., Díaz, J.C.: Tracking moving optima using kalman-based predictions. Evol. Comput. 16(1), 1–30 (2008)
[91] Ryan, C.: The degree of oneness. In: Proc. 1st Online Workshop on Soft Computing, pp. 43–49 (1996)
[92] Salomon, R., Eggenberger, P.: Adaptation on the evolutionary time scale: A working hypothesis and basic experiments. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (eds.) AE 1997. LNCS, vol. 1363, pp. 251–262. Springer, Heidelberg (1998)
[93] Simões, A., Costa, E.: Memory-based chc algorithms for the dynamic traveling salesman problem. In: Proc. 2011 Genetic and Evol. Comput. Conf., pp. 1037–1044 (2011)
[94] Simões, A., Costa, E.: An immune system-based genetic algorithm to deal with dynamic environments: Diversity and memory. In: Pearson, D.W., Steele, N.C., Albrecht, R. (eds.) Proc. 2003 Int. Conf. on Neural Networks and Genetic Algorithms (ICANNGA 2003), pp. 168–174 (2003)
[95] Simões, A., Costa, E.: Improving memory's usage in evolutionary algorithms for changing environments. In: Proc. 2007 IEEE Congr. Evol. Comput., pp. 276–283 (2007)
[96] Simões, A., Costa, E.: Evolutionary algorithms for dynamic environments: Prediction using linear regression and markov chains. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 306–315. Springer, Heidelberg (2008)
[97] Simões, A., Costa, E.: Improving prediction in evolutionary algorithms for dynamic environments. In: Proc. 2009 Genetic and Evol. Comput. Conf., pp. 875–882 (2009)
[98] Singh, H.K., Isaacs, A., Nguyen, T.T., Ray, T., Yao, X.: Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 3127–3134 (2009)
[99] Stanhope, S.A., Daida, J.M.: Genetic algorithm fitness dynamics in a changing environment. In: Proc. 1999 IEEE Congr. Evol. Comput., vol. 3, pp. 1851–1858 (1999)
[100] Tinos, R., Yang, S.: Continuous dynamic problem generators for evolutionary algorithms. In: Proc. 2007 IEEE Congr. Evol. Comput., pp. 236–243 (2007)
[101] Tinós, R., Yang, S.: An analysis of the XOR dynamic problem generator based on the dynamical system. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 274–283. Springer, Heidelberg (2010)
[102] Toffolo, A., Benini, E.: Genetic diversity as an objective in multi-objective evolutionary algorithms. Evol. Comput. 11(2), 151–167 (2003)
[103] Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Proc. 1999 IEEE Congr. Evol. Comput., vol. 3, pp. 1843–1850 (1999)

[104] Ursem, R.K.: Multinational GA optimization techniques in dynamic environments. In: Proc. 2000 Genetic and Evol. Comput. Conf., pp. 19–26 (2000)
[105] Ursem, R.K., Krink, T., Jensen, M.T., Michalewicz, Z.: Analysis and modeling of control tasks in dynamic systems. IEEE Trans. Evol. Comput. 6(4), 378–389 (2002)
[106] Uyar, A.S., Harmanci, A.E.: A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments. Soft Comput. 9(11), 803–814 (2005)
[107] Vavak, F., Fogarty, T.C., Jukes, K.: A genetic algorithm with variable range of local search for tracking changing environments. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 376–385. Springer, Heidelberg (1996)
[108] Vavak, F., Jukes, K., Fogarty, T.C.: Learning the local search range for genetic optimisation in nonstationary environments. In: Proc. 1997 IEEE Int. Conf. on Evol. Comput., pp. 355–360 (1997)
[109] Vavak, F., Jukes, K.A., Fogarty, T.C.: Performance of a genetic algorithm with variable local search range relative to frequency for the environmental changes. In: Proc. 3rd Int. Conf. on Genetic Programming, pp. 602–608 (1998)
[110] Wang, Y., Wineberg, M.: Estimation of evolvability genetic algorithm and dynamic environments. Genetic Programming and Evolvable Machines 7(4), 355–382 (2006)
[111] Weicker, K.: An analysis of dynamic severity and population size. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 159–168. Springer, Heidelberg (2000)
[112] Weicker, K.: Evolutionary algorithms and dynamic optimization problems. Der Andere Verlag (2003)
[113] Weicker, K.: Analysis of local operators applied to discrete tracking problems. Soft Comput 9(11), 778–792 (2005)
[114] Weicker, K., Weicker, N.: On evolution strategy optimization in dynamic environments. In: Proc. 1999 IEEE Congr. Evol. Comput., vol. 3, pp. 2039–2046 (1999)
[115] Woldesenbet, Y.G., Yen, G.G.: Dynamic evolutionary algorithm with variable relocation. IEEE Trans. Evol. Comput. 13(3), 500–513 (2009)
[116] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1(1), 67–82 (1997)
[117] Xing, L., Rohlfshagen, P., Chen, Y., Yao, X.: A hybrid ant colony optimisation algorithm for the extended capacitated arc routing problem. IEEE Trans. Syst., Man and Cybern., Part B: Cybern. 41(4), 1110–1123 (2011)
[118] Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Proc. 2005 Genetic and Evol. Comput. Conf., pp. 1115–1122 (2005)
[119] Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., et al. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 788–799. Springer, Heidelberg (2006)
[120] Yang, S.: A comparative study of immune system based genetic algorithms in dynamic environments. In: Proc. 2006 Genetic and Evol. Comput. Conf., pp. 1377–1384 (2006)
[121] Yang, S.: On the design of diploid genetic algorithms for problem optimization in dynamic environments. In: Proc. 2006 IEEE Congr. Evol. Comput., pp. 1362–1369 (2006)
[122] Yang, S.: Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evol. Comput. 16(3), 385–416 (2008)

[123] Yang, S., Jiang, Y., Nguyen, T.T.: Metaheuristics for dynamic combinatorial optimization problems. IMA J. of Management Mathematics (2012), doi:10.1093/imaman/DPS021
[124] Yang, S., Jin, Y., Ong, Y.S. (eds.): Evolutionary Computation in Dynamic and Uncertain Environments. Springer, Heidelberg (2007)
[125] Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)
[126] Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. Evol. Comput. 12(5), 542–561 (2008)
[127] Yu, E.L., Suganthan, P.N.: Evolutionary programming with ensemble of explicit memories for dynamic optimization. In: Proc. 2009 IEEE Congr. Evol. Comput., pp. 431–438 (2009)
[128] Zeng, S., Shi, H., Kang, L., Ding, L.: Orthogonal dynamic hill climbing algorithm: ODHC. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 79–105. Springer, Heidelberg (2007)
[129] Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Prediction-based population reinitialization for evolutionary dynamic multi-objective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 832–846. Springer, Heidelberg (2007)
[130] Zou, X., Wang, M., Zhou, A., Mckay, B.: Evolutionary optimization based on chaotic sequence in dynamic environments. In: Proc. 2004 IEEE Int. Conf. on Networking, Sensing and Control, vol. 2, pp. 1364–1369 (2004)