Analyzing the Impact of Evolutionary Algorithms on Optimization Problems

Suman Rani Yadav[1], Pramila Chandel[2] and Bhawna Kaushik[3]
[1]*Assistant Professor, Lloyd institute of Engineering and Technology, suman.yadav@liet.in*
[2]*Assistant Professor, Lloyd institute of Engineering and Technology, Pramila.chandel@liet.in*
[3]*Assistant Professor, Lloyd institute of Engineering and Technology, Bhawna.kaushik@liet.in*

Abstract

Evolutionary algorithms (EAs) are essential tools for tackling complex optimization problems, especially in dynamic environments where objective functions and constraints evolve over time. These bio-inspired algorithms draw from Darwinian principles to continuously adapt and find optimal solutions. This paper reviews the state of the art in EAs, focusing on their application to dynamic optimization problems (DOPs), including benchmark problems, performance measures, and algorithmic techniques. The study highlights the strengths and weaknesses of current methods, identifies gaps in research, and explores future directions. Key areas covered include evolutionary computation, genetic programming, and genetic algorithms, along with specific strategies for change detection and diversity maintenance in dynamic settings.

Keywords: Evolutionary Algorithms, Dynamic Optimization, Benchmark Problems, Genetic Algorithms, Performance Metrics, Change Detection, Diversity Introduction, Multi-Objective Optimization

1.INTRODUCTION

Evolving algorithms are crucial for tackling complicated optimization problems to their full potential [1–5]. Because of the inherent complexity of problems involving dynamic environments, bio-inspired algorithms have developed out of the concept of Darwinian evolution. In an effort to discover the best solutions to multi-objective problems, researchers are constantly sifting through data in the real world. These environmental changes impact the problem's objective function, the problem's instance, and the environmental restrictions that arise on the problem's battlefield [4, 6, 7]. Optimisation challenges aim to find optimum solutions and track their progress as a function of the onset of change [8]. Nothing is ever really final, yet evolutionary algorithms have shown to be very effective in solving real-world issues. These methods might be difficult to use when trying to solve multi-objective optimization issues due to their large number of parameters [9].

Our contribution to this study is to review the various settings in order to draw conclusions. Discovering the conditional search that encompasses all elements inside the surroundings is of great importance to us. Research into dynamic environments is ongoing, and there has been a great deal of work done in this area, including the development of several methods and techniques. In our study, we have also explored the case of an imperfect environment. Our primary goal is to delve deeply into the topic of optimization in dynamic environments, covering topics such as benchmark problems and solvers, EA performance in various environments, algorithm approaches, real-world applications, algorithm strengths and weaknesses, and theoretical investigation of various algorithms. Our aim is to go over the following points: first, how the current methods are functioning (we've explained this in detail), secondly, the assumptions made by these methods, their strengths and weaknesses, any gaps in the current methods, and finally, the difficulties and potential solutions that may be solved by conducting future research on evolutionary algorithms in various settings.

Nowadays, optimization issues may be found in every area of business, academia, and management. We may use a real-world issue as an example of optimization. Making the most efficient use of the resources at hand while keeping costs low and quality high is one such example. Optimisation from every angle is necessary in almost every commercial, building, and technological project. There are a great deal of optimization issues within the framework of evolutionary algorithms. Schedule optimization utilizing the ant colony method on non-ideal iris image segmentation, routing challenges, and traveling salesperson problems are prevalent among most of these issues, as are NP-hard problems, for which no solution is achievable in polynomial time. [10, 14], [19], [21].

An enhanced grouping genetic algorithm was suggested by Shahzadeh et al. [20]. This model outperforms the previous models and resolves issues related to clustering algorithms. This model presents and tests an enhanced

clustering method, the results of which demonstrate high performance and the attainment of the ideal solution. Compared to earlier outcomes, this one is much superior.

2. Evolutionary Algorithms

In essence, evolutionary algorithms are meta-heuristics based on populations, which often have several solutions rather than just one. Numerous biological experiences, each with its own unique set of governing principles, make up nature. Based on his work with plant and animal evolutionary systems, Charles Darwin proposed the concept of biological algorithms. Nature and environmental behavior provide as inspiration for evolutionary algorithms [6], [7]. Function optimization with high-dimensional data and global optimization issues are two common areas where evolutionary techniques find use. It is based on the idea of species populations and draws inspiration from biological evolutionary theory. The field of soft computing is a source of inspiration for evolutionary algorithms. Evolutionary algorithms are mostly used to address adaptive non-linear situations. When it comes to optimization difficulties, their learning quality is top-notch. They are recognized by quick and explicit optimized algorithms for their robustness. Convergence is quite effective and has a high prediction accuracy, albeit it varies from issue to problem. The whole evolutionary process revolves around the central emphasis of strategy creation and optimization. Typically, an evolutionary algorithm will have three main components: a population at time t, operators for variation and selection [20].

2.1 Evolutionary Computation

Using the principles of Darwinian natural biological evolution theory as its foundation, evolutionary computation may tackle both simple and complicated optimization problems with many objectives. Both groups believe in the concept of populations as collections of people, with current populations using reproductive operators like mutation and crossover to create new generations. They breed better people from older populations, and the resulting new residents are superior in every way. Recombination, a process that mixes at least two people from a population to create new offspring, allows us to build upon prior solutions by creating new ones. A fitness notion and fitness function are the backbone of evolutionary algorithms. Within a population, only those people with a high fitness value will be able to live, while those with a low fitness value would perish. Researchers have created a plethora of surveys and applications in recent years, with a focus on addressing issues with complex optimization, combinatorial optimization, multi-objective optimization, data mining, and imperfections in dynamic and imperfect environments. Other approaches are capable of self-adaptation as well. Algorithms are considered self-adaptive if they can and often do regulate their parameters independently [2].

2.2 Evolutionary Strategies

When it comes to optimizing continuous functions, evolutionary algorithms provide a number of methods, one of which is evolutionary strategies. In this respect, there are two steps: include the parent in the selection process and excluding it. Mutation was the first evolutionary strategy, and most organisms were considered as individuals. The primary emphasis of evolutionary methods is on individual behavior [2].

2.3 Evolutionary Programming

Evolutionary algorithms also make use of evolutionary programming. The fundamental goal of every evolutionary algorithm is learning, and the main reason for presenting this method is to construct optimization techniques for evolutionary games and sequence prediction. During training, these algorithms optimize various parameters to perform a certain function. A important use of evolutionary programming is in training artificial neural networks. Since the crossover operator genetic algorithm avoids using evolutionary programming, it is primarily concerned with studying species behavior [2].

2.4 Mechanism of Genetic Programming

Another population-based meta heuristic method that operates in a tree-derivative way is genetic programming. Operators, operands, terminal nodes, and leaf nodes make up GP. Terminals are the program's variables and constants in genetic programming. To begin using GP, one must start a population of people; typically, this is done at random. Following the population's initialization in GP, subsequent generations undergo evolution based on Darwinian evolution, mutations, and crossovers. Probabilistic selection is then used to create new individuals for the population. Genetic operators then begin to work and generate offspring in the subsequent generations. It is possible to create a population with a wide range of genetic variations by combining different

types of cross-over and mutations. Utilizing the principle of "survival of the fittest" to identify the most effective solutions within a given domain, GP has found widespread use in machine learning as a central solver for data mining challenges.

2.5 Genetic Algorithm

When it comes to soft computing, genetic algorithms are among the most revolutionary tools. For many multi-dimensional problems, it is the most popular populations-based evolutionary algorithm. It is a randomized optimization method that employs searching strategies in its expanded population space; nonetheless, basic GA, being relatively general, does not perform well in imperfect and dynamic contexts. In order to solve challenging optimization issues, it draws inspiration from biological intelligence. Consisting of solutions, it forms a population, and chromosomes are the building blocks of that population [24].A genetic method that is based on grouping and has a better clustering model is suggested in [20]. The results demonstrated a significant difference between GGA and IMGGA, the grouping genetic algorithm. IMGGA outperformed GGA in 70% of situations [20]. The idea at its core is the principle of natural selection at work in populations. To generate new generations from a set of parents' genes, genetic algorithms employ two reproduction operators: crossover and mutation. Similar to other evolutionary algorithms, this one uses the fitness function to decide which solutions or individuals will survive. Different types of crossover are important reproduction operators. Multiple goal optimization, distributed problem solving, and a host of other real-world applications have found a home in genetic algorithms, which now have several varieties and are being further refined by researchers. cited as[21],[22],[23], and [24].

3. Dynamic optimization problems

This article devotes a significant amount of space to discussing every facet of dynamic optimization issues. The area of dynamic optimization has seen a great deal of research. When it comes to addressing difficult optimization issues, evolutionary computations and swarm intelligence algorithms are great options. These algorithms aim to find the ideal answer at certain intervals and keep track of it in search space. We define dynamic optimization problems as those where the landscape changes to find new optimal solutions, and the optimum values change from the beginning to the end of the time interval. This is because of their inherent evolving power and learning capability.

Since this area of study is still in its infancy, with research having only advanced over the past quarter of a century and more, it is fair to say that it is relatively young. Books, journal issues (such as IEEE Transactions on Evolutionary Computations and Soft Computing), and conferences (such as the IEEE Congress on Evolutionary Computations) devoted to the topic abound. This area has been the subject of hundreds of PhD dissertations. Uludug et al. [8] suggested a population framework that incorporates offline and online learning, and Crus et al. [9] conducted a thorough survey of optimizing dynamic problems, issues within them, and performance measurements in all of them. A large amount of work has been done in the field of static and dynamic environments, including technical developments, literature studies, and surveys. Some of this work has focused on synthetic dynamic problems, such as function optimization. Optimization using linear regression models is the focus of Eriksson and Simoes's [55] research. The use of differential evolution in solving dynamic optimization problems was discussed in Comelius's PhD thesis [10]. In a related work, Brank et al. [11] proposed an evolutionary optimization algorithm for uncertain environments. Montazeri et al., 2013 [21] investigated the impact of ant algorithms on non-ideal iris image segmentation. They found that merging strong image segmentation with a metaheuristic algorithm improved accuracy by optimizing iris borders, reducing selected features, and localizing ACO. The system achieved the lowest number of features, cost, and accuracy by comparing the results produced by incorporating image processing with a metaheuristic optimization technique. There are still several places where this strategy might be improved. A number of researchers have suggested evolutionary gaming strategies for solving problems, such as a checkers game (Amold et al., 2012; Amold et al., 2013). Others have suggested methods for dealing with noisy environments and scheduling dynamic optimization problems using genetic algorithms. Brank et al. [39] presented an approach for usage in dynamic situations that optimizes the fitness function inside the solution to provide a resilient solution, whereas Kluwer et al. [29] presented a study on swarm optimization applied to such settings. Some memory-based methods also operate in dynamic settings, such as those suggested by [41], [42] for an optimized dynamic environment, and by Brank et al. [40] for an application in internet cache routes utilizing a dynamic optimization approach. In order to address dynamic optimization issues, Brank et al. [48] suggested a method based on several populations, and Brank [50] put forth an evolutionary algorithm. In order to determine the average and standard deviation of product quality, Zabihinpour et al., 2014 [51] suggested an optimization strategy based on fuzzy logic. This strategy involves building fuzzy control charts using fuzzy numbers in the shape of triangles. Its foundation in the sample mean gives it two benefits: one, it keeps track of process information; second, it's a tool for controlling the process via

process adjustment. There is a need for further study, although studies have shown that the suggested method performs better.

In contrast, numerous works have been developed for practical use; for example, Akbar et al. [54] suggested a face recognition model that combines hybrid space features with support vector machines; this model is an intelligent approach to soft computing recognition problems; and in order to acquire high-quality features, the authors used a variety of algorithms operating as learner classifiers. By extracting features from the dataset using both transformation and local pattern based strategies, we can evaluate the performance of the suggested model. The features are extracted using a variety of techniques, evaluated using 10-fold cross validation, and then utilized to obtain an accuracy of 92.1% using SVM. Cantu-Paz [45] made a contribution to evolutionary robotics, Bumham [44] worked on aeronautical design, and Bull et al. [43] presented a model framework in evolutionary computing that is utilized to tackle dynamic optimization issues. A knapsack-based method for solving combinatorial optimization problems has been suggested by Carlisle [46], while a method for solving multi-objective optimization in dynamic contexts has been provided by Branke [48]. In noisy and unpredictable settings, Goh [49] lays out a method for self-adaptation and mutation; Brank [11] provides an overview of such settings and other state-of-the-art methods that have been established thus far. For the purpose of solving combinatorial optimization issues, Branke [50] suggested a dynamic benchmark function generator.

Branke [28] presented an indexing evolutionary algorithm that relies on the idea of a memory-based dynamic evolutionary algorithm; Simeo [59] suggested a genetic algorithm for dealing with complicated problems in dynamic environments; and Branke [29] put forward an associative memory model for use in such settings, wherein new individuals are generated using data collected from the surroundings. Their adaptive model, as suggested in [38], develops evolutionary strategies and then adjusts them based on their surroundings. In [61], for example, the authors suggested using cultural algorithms for dynamic optimization. Presented a cultural method for solving dynamic optimization problems using a rule-based framework in [62], Cultural algorithms were suggested by Rossi et al. [28] for use in optimizing functions in dynamic settings, influencing environmental dynamics, learning how to adapt to changes in the environment, and developing strategies for such environments. Even in dynamic settings, Richter et al. [56] achieved excellent results by optimizing functions. The optimization of multi-modal functions, networks, and vehicles, as well as the study of market stocks and traveling salespeople, are just a few other areas that make use of dynamic settings.

Previous surveys on dynamic optimization problems cover all the bases in terms of the work done and the methods used, but there's always been a hole. We've filled that hole by providing a comprehensive review of the current approaches, explaining why they fail in certain cases and outlining the difficulties faced by the field. We've also laid out the advantages and disadvantages of optimization problems in great detail. In this overview, we have concentrated on several facets of optimization issues, including benchmark problems, performance metrics, algorithmic and practical approaches, and many more.

4. Benchmarking issues

In order to tackle optimization issues in dynamic environments and test their performance with varied settings, Branke et al. [42] developed the moving peak benchmark. To allow for the testing of algorithms on a variety of situations and the evaluation of dynamic environments across a range of parameters and settings, the benchmark is made up of several moving peaks. Optimal points, peak height, and form are all features of the moving peak function in a multi-dimensional problem space [10].

For an n-dimensional moving peak benchmark, the following parameters are defined: n-peak count, dimensions, peak height and width at their maximum and minimum values, change period, change severity, change frequency, function, and correlation. In the moving peak benchmark issue, we may modify the magnitude and height of the change by adjusting the parameters in each of the three settings. All three of these possibilities were proposed by Brank et al. [39]. Scenario 2 has been the most utilized in shifting peak benchmarks from the beginning, therefore most researchers employ it in their work [27–40].

4.1 Contemporary Benchmark Methodologies

By analyzing the existing literature, we investigate the characteristics and uses of numerous cutting-edge benchmark problems. These issues have thus far been generated. A vast number of benchmark problems enable the resolution of scheduling challenges, combinatorial optimization issues, and multi-modal function optimization, among many other valid concerns. Benchmark issues are categorized based on the frequency of

occurrence in which they are encountered. The features encompass a wide range of classification criteria. These include the time dependence of the algorithm on the benchmark, the predictability of the benchmark with respect to whether the solution it predicts has superior optimums or not, the visibility of modifications implemented in the optimization algorithm, the quantity of constraints that impact those modifications over time, the overall number of objectives, the nature of the modifications involved, and the factors that influence those modifications.

There are a variety of benchmarks that have been done in the past, each with its own unique characteristics. For example, while most benchmarks do not depend on time, there are a few that do, as seen in [64], [65]. Some benchmarks make change detection easy, while others are more challenging, as seen in [69], [70]. A change in the objective function has occurred in many benchmark problems; however, in some instances, such as in [66], [67], and [68], the constraints have also changed. While most of the current state-of-the-art benchmark challenges focus on addressing optimization problems with a single goal, there are a few that include dynamic multi-objective optimization techniques, such as in [75], [76], and [77].

Table I. Benchmark Generators in Continuous Problems

Functions	**Changes Predictable?**	**Changes detection by using few detectors?**	**Single/Multi Obj?**	**Changes are cyclic, periodical and recurrent?**	**s that change s**		
				Objective functions	**Variables Domain**	**Variables Numbers**	**Constraint functions**
Switching Functions [69]	Mostly No	Yes & no	Single	Yes	Not Mention	No	No	No
Moving Peaks [63]	Mostly No	Yes	Single	Configurable	Yes	No	No	No
Oscillating Peaks [63]	Mostly No	Yes	Single	Yes	No	No	No	No
DF1 [80,81]	No	Yes	Single	No	Yes	No	No	No
Gaussian Peaks [82]	No	Yes	Single	No	Yes	No	No	No
Disjoint Landscapes [70]	Mostly No	pends on Number of Peaks	Single	Yes	Yes	No	No	No
Dynamic Rotation [83]	Mostly No	Partly Detectable	Single	Yes	Yes	No	No	No
MOO Generator [75]	Yes	Yes	Both	Not	Yes	No	No	No
FDA [76], ZJZ [78], HE [77]	Configurable	Yes	Multi Obj	Yes	Yes	No	No	No
Dynamic Test [84]	Partly	Yes	Single	Yes	Yes	No	No	No
CDOPG XOR [85]	No	Yes	Single	Yes	Yes	No	No	No
CEC 09 [71]	No	Yes	Single	Yes	Yes	No	No	No
G24 Constrained Set [66, 67]	Yes	No	Single	Yes	Yes	No	No	No

Dynamic Constrained [68]	No	No	Single	Partly	Yes	No	No	No

5. Performance Measures in Optimization Problems

When it comes to optimization issues, whether they be imperfect evolutionary or dynamic, performance assessment is a key component. Here, we've broken down the current methods, pointed out their advantages and disadvantages, and talked about ways to fix the problems these methods have. We can further categorize these methods into two sets: those that focus on optimization performance and those that deal with problem behaviors. Here are the two requirements that are outlined below:

5.1 Utilization of Performance Metrics in Optimization Problems

One way to approach optimization issues is to consider the algorithms' optimality and assess them accordingly. One of the metrics included in them might be a fitness function or an objective function. The global optimums are determined by the distance measure, and the best solutions are always found there. Subsequently, performance metrics are categorized as follows:

A) Each optimization method requires several runs to determine the best potential performance of its curves, which increases the total number of generations. To start, this metric has been used before by [32], [33], [27], [34], and [35]. This metric is still used by researchers since it is one of the best at capturing the true and practical performance of optimization techniques. Reason being, this metric reliably displays the optimal curves in the graph that has been generated after a series of algorithmic runs. In terms of the amount of runs, performance could differ between algorithms.

B) When it comes to optimization algorithms, the majority of mistakes happen both offline and online [67], [68]. The number of generations is used to determine the average error, which is used to evaluate performance. Calculations account for the majority of internet mistakes. In order to get improved performance in optimization issues, several researchers have made modifications to offline and online faults, as shown in [86].

C) Another way to quantify error is to do it before environmental changes happen. This method, as suggested in [70], involves taking the difference between the best and most ideal solutions and averaging them to get the error. Situations where we wish to reach the point of mistake before the issue changes are ideal for this. We then evaluate this mistake in comparison to several optimization strategies. Nevertheless, after reviewing this, we discovered the following drawbacks. First, it doesn't calculate or disclose the algorithm's current performance; second, the error computation doesn't seem to be normalized on certain scales, which might lead to a bias in favor of large-number mistakes. Finally, knowing the global optimum value at the time of change is necessary for this error approach.

5.2 Performance Metrics Predicated on Algorithm Behaviors

For optimization issues addressed by evolutionary algorithms, this is the secondary metric for performance. A key characteristic of evolutionary optimization algorithms is their ability to preserve diversity through a maximum number of runs, which in turn helps to maintain the fitness of the solution. However, this property is shared by all optimization algorithms. As part of evaluating performance, all of these metrics are crucial. What follows is an explanation of the primary behaviors:

A) A focus on variety is one of the behaviors of optimization algorithms; quantifying diversity in environments is essential when dealing with algorithms that behave like evolutionary heat. The literature contains numerous diversity-based metrics, including the most significant and widely employed distance measure, entropy (41), the hamming distance (42), [43], and [44], the coverage of peaks in graphs (81], and the maximum spread coverage in search space (46]).

The Hamming distance, which is widely recognized as a diversity metric in optimization issues, was initially computed as the separation between every individual in the population and the global optimums (as proposed in reference [42]). However, subsequent research has proposed an alternative approach wherein the distance is determined using the nest-only individuals of the population (as detailed in [43]). The moment of inertia [45], a

statistical metric founded on the concept of inertia in physics, has been employed by researchers to quantify the diversity of evolutionary algorithms. In relation to search space, the proposed inertia is conceptually comparable to the hamming distance measurement; however, it surpasses the hamming distance measurement in terms of efficiency and computational expense. An additional method for assessing diversity is peak cover [77], which employs the total number of peaks to evaluate algorithmic behavior and landscapes that require comprehensive information on peaks.

B) Performance degradation after an update is the second characteristic of optimization methods. Whenever a modification is introduced into certain dynamic optimization techniques, performance lowers. There is a decline in performance when considering fitness, which led to the proposal of a stability technique [37] that uses algorithmic accuracy to assess fitness. Using this approach, we can determine the amount of fitness lost due to environmental changes.

5.3 Evaluation of Performance in Multi-Objective Optimization

The performance of dynamic MOO problems has been the subject of several research. In multi objective optimization problems, for example, there are various criteria that may be used to quantify performance at certain intervals, as suggested in [46], [48], and [49]. In multi-objective optimization issues, accuracy is a key metric. For example, in [51], the ratio of the present volume to the maximum volume obtained and vice versa was specified. In optimization problems with several objectives, we may determine stability and performance using this accuracy metric.

5.4 Some Research Issues/Questions on Optimization Problems

We have extensively examined the factors that influence performance and behavioral metrics in dynamic optimization problems in previous conversations. A number of inquiries and apprehensions are prompted by evolutionary dynamic optimization. Does the primary objective of dynamic optimization problems, for instance, appear to be the attainment of global optima? It is debatable whether performance metrics designed for optimization problems accurately represent the required metrics. A number of studies, including [33], [17], and [43], have put forth metrics for assessing the performance of optimization problems. An attempt is made in [33] to address inquiries regarding the meaning and requirements of optimization problems in the real world.

Another area that needs further investigation is the optimality measures used in optimization problems. While many dynamic optimization problems only consider absolute fitness values, relative fitness values are just as important and can be utilized. By comparing them with different algorithms, intriguing results may emerge. Some accuracy metrics have attempted to standardize the fitness value on a given scale in [37]. Thus, data on changes in the maximum and minimum period intervals is required in order to have such a normalizing factor.

Thirdly, there is no guarantee that past findings will correspond with future ones when using behavioral metrics to evaluate optimization problem performance. Results from [47] demonstrate that behavioral stability is more than just a solution quality metric [52]. In dynamic optimization issues, the most important factor is the link between behavioral and performance measurements.

6. Techniques Used for Optimization

We have covered the dynamic evolutionary optimization algorithms, current research, and the pros and cons of all the newly created methods in this area.

6.1 Evolutionary Algorithms Goal in Optimization Problems

The primary goal in a static environment is to find the optimal solution as quickly as possible. However, when the environment changes and time variations, the optimal solution constantly moves, and optimization algorithms draw on their prior knowledge to find the local optimal solution. We have covered many dynamic optimization strategies, along with their advantages and disadvantages, and discovered methods that can swiftly adapt to changing situations.

6.2 Change Detection in Dynamic Optimization Problems

As mentioned earlier, change detection is quick in some dynamic environments but sluggish in others. We have classified change detection as either detecting changes by reevaluating solutions or detecting changes based on algorithm behaviors.

A) Some current approaches for obtaining best solutions are memory based approaches [56] and re-evaluation of previously evaluated solutions [53], [54], [55]. In this technique, the algorithm is constantly busy evaluating solutions that gave better fitness, so in order to better learn them, it re-evaluates them to detect potential changes in the environment. Altering the population at a single location, as in [57], combining numerous random solutions in the populations, as in [58], [59], or finding solution peaks, as in [60], [61], are all alternatives to search population based procedures.

B) Change detection is already a computationally intensive process; adding detectors to it would require additional functional calculations; so, the ideal strength of detectors is necessary to increase the algorithm's performance. This is one of the method's disadvantages. For the most part, current methods use a smaller number of change detectors to cut down on functional computations. However, when search spaces undergo changes, as in [66], [67], and [85], using a smaller number of change detectors may not be enough to ensure optimal change detection in these environments. The following are examples of previous efforts to address this issue: [22], [23], and [26]. There is a clear demonstration in [26] that this re-evaluation method will be of great assistance if change detection is going to be challenging, so now the size and computational complexity of the detector are the most important factors. Another drawback of this approach is the constant need to reevaluate detectors at each phase, which adds unnecessary expense with each generation.

Another potential drawback is that this re-evaluation process could not work well with noisy fitness function issues; noise has the potential to mislead the algorithm and defeat the aim of change detection.

6.3 Detection of Changes Predicated on the Actions of Algorithms

In order to determine the optimal solution, we have already specified the characteristics of the optimization algorithms that we will be using. Change detection relies on keeping an eye out for changes, according to swarm intelligence literature such that found in [62]. According to [22], one way to tell when things are changing in the environment is by looking at how diverse the behaviors are and how they relate to the fitness value.

One benefit of this behavioral model is that it doesn't need any additional calculations. However, because it doesn't utilize a change detector, it can't guarantee that it will always detect changes [26]. This method might have an out-of-the-ordinary reaction rate and produce false positives, similar to what happened in [62]. The fact that this can be algorithm-specific when it comes to change detection is another potential drawback.

6.4 Diversity Initiation Upon the Arrival of Change

The introduction of variety is fundamental to dynamic optimization problems as new information becomes available. Keeping it up and improving it to a certain degree are so crucial.

A) It is difficult to keep track of the optimal point in a dynamic fitness landscape because it is constantly changing area and location; our main goal is to slowly move from the objective towards the global optima. In a static environment, finding the optimal point is the main success metric. However, in a dynamic optimization setting, convergence varies according to the arrival of change. One possible answer is to increase variety, but it's also possible that this won't give us the best outcomes. Variable search in local optima is mostly discussed in [63], [64], and a feature partitioning approach is also suggested in [65]. There was a proposal for an adaptation to genetic programming in [66] that would raise the mutation rate, decrease elitism, and, after the adjustment, raise the likelihood of crossover. A population's ability to adapt to new circumstances depends on its ability to exploit the central principle of diversity introduction in multi-objective optimization problems. After a change has occurred, swarm intelligence also enjoys adding variety. In order to re-diversify swarms in the search space and create variation in the change situation, a randomization mechanism has been suggested in [53].

B) Pros and cons of diversity introduction upon change's arrival: Both preserving and improving variety are central to the strategies put out by this school of thought. Because it is able to zero down on the search process and respond instantly to changes, it has a distinct advantage. When dealing with scenarios where the amount of

change is large, yet optimums can be monitored independently, as in [65], [63], these strategies work nicely. On the flip side, there are a few drawbacks to consider.

7. Conclusion

In conclusion, evolutionary algorithms have proven to be highly effective for solving complex optimization problems in dynamic environments, leveraging their adaptive and learning capabilities. The study provides a thorough review of the current methods, benchmarks, and performance metrics used to assess these algorithms. It highlights the strengths and limitations of various approaches, including evolutionary computation, genetic programming, and genetic algorithms. Furthermore, the paper underscores the importance of change detection and diversity introduction techniques in maintaining algorithm performance in dynamic contexts. Despite the progress made, there are still significant research gaps and challenges to address, particularly in optimizing multi-objective problems and improving the robustness of EAs in noisy and unpredictable environments. Future research should focus on enhancing algorithmic strategies, developing more comprehensive benchmarks, and refining performance metrics to ensure that evolutionary algorithms can effectively adapt to the complexities of real-world dynamic optimization problems.

REFERENCES

[1] P. H. Winston, Artificial Intelligence. Reading, Massachusetts: Addison-Wesley. 1984.

[2] D. G. Bobrow, Artificial Intelligence in Perspective. The MIT Press, 1994.

[3] A.P. EngelBrecht, "Evolutionary Computations & Swarm Intelligence", in Computational Intelligence an Introduction, London, UK: John Wily & Sons Inc., 2002

[4] K.A. De Jong, "Introduction", in Evolutionary Computations a Unified Approach, London, UK: MIT Press, 2006

[5] A. Konar, Computational Intelligence Principles, Techniques and Applications. Springer-Verlag 2005

[6] G. Kendall, and Y. Su, "Imperfect Evolutionary Systems," IEEE Transactions on Evolutionary Computations, vol. 11, no. 3, pp. 294–307, 2007.

[7] G. Kendall, and Al. Khateeb, "Introducing Individual and Social Learining into Evolutionary Checkers," IEEE Transactions on Evolutionary Computations, vol. 4, no. 4, pp. 258–269, 2012.

[8] G. Uludug, B. Kiraz, A.Sima and E. Ozcan, "A hybrid multi population framework for dynamic environments combining online and offline learning," Soft Computing, vol. 17, no. 12, pp. 2327–2348, 2013.

[9] C. Cruz, J.Gonzalez and D. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures." Soft Computing, vol. 15, no. 12, pp. 1427–1448, 2011.

[10] M. Cornelius, "Adaptive Multi Population Different Evolution for Dynamic Environments," Ph.D. dissertation, Univ. Pretoria,, 2012.

[11] Y. Jin, and J. Branke, "Evolutionary optimization in Uncertain Environments-A Survey," IEEE Transactions on Evolutionary Computations, vol. 9, no. 3, pp. 303–317, 2005.
109

[12] D. V. Arnold, Noisy Optimization with Evolution Strategies. Norwell, MA: Kluwer, 2002.

[13] D. V. Arnold and H.-G. Beyer, "Efficiency and mutation strength adaptation of the ES in a noisy environment," in Parallel Problemsolving from Nature. ser. LNCS, M. Schoenauer et al., Eds. Berlin, Germany: Springer-Verlag, 2000, vol. 1917, pp. 39–48.

[14] NRC, United States National Research Council, Developments in Artificial Intelligence. Funding a Revolution: Government Support for Computing Research. National Academy Press. 1999

[15] R.Jensen, Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches, Wiley-IEEE Press 2008.

[16] A.E. Eiben, and J.E. Smith, Introduction to Evolutionary Computing. Natural Computing Series, 2nd edition, 2007. [17] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley, 1989.

[18] J. Johnston, The Allure of Machinic Life: Cybernetics, Artificial Life, and the New AI. MIT Press, 2008

[19] D. Goodman and R. Keene, Man versus Machine: Kasparov versus Deep Blue. Cambridge. MA: H3 Publications, 1997.

[20] S. Fazeli and Barkhordari, "An Improved Grouping Genetic Algorithm", J. Appl. Environ. Biol. Sci., 6(1) 38-45, 2016.

[21] Montazeri, Mehdi Zadeh and Mohammad Mosoud "Application of Ants Colony Algorithm for Non ideal Iris Image Segmentation", . J. Appl. Environ. Biol. Sci., 3(12) 30-33, 2013.

[22] C. Bierwirth and H. Kopfer, "Dynamic task scheduling with genetic algorithms in manufacturing systems," Dept. Econ., Univ. Bremen, Bremen, Germany, Tech. Rep., 1994.

[23] C. Bierwirth and D. C. Mattfeld, "Production scheduling and rescheduling with genetic algorithms," Evol. Comput., vol. 7, no. 1, pp. 1–18, 1999.

[24] J. A. Biles, "Genjam: A genetic algorithm for generating jazz solos," in Proc. Int. Comput. Music Conf., Aarhus, Jutland, Denmark, 1994, pp.131–137.

[25] T. Blackwell and I. Branke, "Multiswarm optimization in dynamic environments," in Applications of Evolutionary Computing.
ser. LNCS, G. R. Raidl et al., Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3005, pp. 489–500.

[26] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in Proc. Genetic Evol. Comput. Conf., W. B. Langdon et al., Eds., 2002, pp. 19–26.

[27] J. Branke, "Creating robust solutions by means of evolutionary algorithms," in Parallel Problem Solving from Nature, ser. LNCS. Berlin, Germany: Springer-Verlag, 1998, pp. 119–128.

[28] J. Branke "Memory enhanced evolutionary algorithms for changing optimization problems," in Proc. Congr. Evol. Comput., vol. 3, 1999, pp. 1875–1882.

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

[30] Brank, J "Reducing the sampling variance when searching for robust solutions," in Proc. Genetic Evol. Comput. Conf., L. Spector et al., Eds., 2001, pp. 235–242.

[31] J. Branke, P. Funes, and F. Thiele, “Evolving en-route caching strategies for the Internet,” in Lecture Notes in Computer Science, vol. 3103, Proc. Genetic Evol. Comput. Conf., K. Deb et al., Eds., 2004, pp. 434–446.
[32] J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck, “A multipopulation approach to dynamic optimization problems,” in Adaptive Computing in Design and Manufacturing 2000, ser. LNCS. Berlin, Germany: Springer-Verlag, 2000.
[33] J. Branke and D. Mattfeld, “Anticipation in dynamic optimization: The scheduling case,” in Parallel Problem Solving from Nature.
ser. LNCS,
[34] M. Schoenauer et al., Eds. Berlin, Germany: Springer-Verlag, 2000, vol. 1917, pp. 253–262.
[35] J. Branke and H. Schmeck, “Designing evolutionary algorithms for dynamic optimization problems,” in Theory and Application of Evolutionary Computation: Recent Trends, S. Tsutsui and A. Ghosh, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 239–262.
[36] J. Branke and C. Schmidt, “Selection in the presence of noise,” in Lecture Notes in Computer Science, vol. 2723, Proc. Genetic Evol. Comput. Conf., E. Cantu-Paz, Ed., 2003, pp. 766–777.
[37] “Sequential sampling in noisy environments,” in Parallel Problem Solving from Nature, ser. LNCS. Berlin, Germany: Springer-Verlag, 2004.
[38] “Fast convergence by means of fitness estimation,” Soft Comput., vol. 9, no. 1, pp. 13–20, 2005.
[39] J. Branke, C. Schmidt, and H. Schmeck, “Efficient fitness estimation in noisy environment,” in Genetic and Evolutionary Computation, L. Spector et al., Eds. San Mateo, CA: Morgan Kaufmann, 2001, pp. 243–250.
[40] J. Branke and W.Wang, “Theoretical analysis of simple evolution strategies in quickly changing environments,” in Lecture Notes in Computer Science, vol. 2723, Proc. Genetic and Evol. Comput. Conf., E. Cantu-Paz et al., Eds., 2003, pp. 537–548.
[41] D. Büche, “Accelerating evolutionary algorithms using fitness function models,” in Proc. GECCO Workshop Learn., Adapt. Approx. Evol. Comput., 2003, pp. 166–169.
[42] D. Büche, P. Stoll, R. Dornberger, and P. Koumoutsakos, “Multiobjective evolutionary algorithm for the optimization of noisy combustion problems,” IEEE Trans. Syst., Man, Cybern., Part C, vol. 32, no. 4, pp. 460–473, 2002.
[43] L. Bull, “On model-based evolutionary computation,” Soft Comput., vol. 3, pp. 76–82, 1999.
[44] K. Burnham and D. Anderson, Model Selection and Multimodel Inference, 2nd ed. Berlin, Germany: Springer-Verlag, 2002.
[45] E. Cantu-Paz, “Adaptive sampling for noisy problems,” in Proc. Genetic Evol. Comput. Conf., 2004, pp. 947–958.
[46] A. Carlisle and G. Dozier, “Tracking changing extrema with adaptive particle swarm optimizer,” in Proc. World Automation Cong., Orlando, FL, 2002, pp. 265–270.
[47] L. A. Zadeh, “Fuzzy sets,” Information and Control, vol. 8, pp. 338–353, 1965.
[48] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer,1394 2001.
[49] C.-K. Goh, K. C. Tan, Evolutionary Multi-objective 1395 Optimization in Uncertain Environments: Issues and Algorithms, Springer Publishing Company, Incorporated, 2009.
[50] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: IEEE Congress on Evolutionary Computation, CEC, Vol. 3, IEEE, 1999, pp. 1875–1882.
[51] S. Mojtaba, Ariffin, Tang S.H and A.S. Azfanizam, “Fuzzy Based Approach for Monitoring the Mean and Range of the Product Quality”,. J. Appl. Environ. Biol. Sci., 4(9) 1-7, 2014.
[52] A. Younes, Adapting evolutionary approaches for optimization in dynamic environments, Doctor of Philosophy (PhD) in Systems Design Engineering, Faculty of Engineering, University of Waterloo, Canada, Faculty of Engineering, University of Waterloo, Canada (2006).
[53] C. L. Ramsey, J. J. Grefenstette, Case-based initialization of genetic algorithms, in: S. Forrest (Ed.), International Conference on Genetic Algorithms, Morgan Kaufmann, 1993, pp. 84–91.
110
[54] Shahid Akbar, Ashfaq Ahmad, Maqsood Hayat and Faheem Ali, “Face Recognition Using Hybrid Feature Space in Conjunction with Support Vector Machine”, J. Appl. Environ. Biol. Sci., 5(7) 28-36, 2015.
[55] A. Simo˜es, E. Costa, Evolutionary algorithms for dynamic environments: Prediction using linear regression and markov chains, in: International Conference on Parallel Problem Solving from Nature, PPSN, Vol. 5199 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, pp. 306–315.
[56] H. Richter, S. Yang, Memory based on abstraction for dynamic fitness functions, in: . Mario Giacobini et al (Ed.), Applications of Evolutionary Computing, Vol. 4974 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, pp. 596–605.
[57] J. Branke, Evolutionary approaches to dynamic optimization problems – introduction and recent trends, in: J. Branke (Ed.), GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2003, pp. 2– 4.

[58] C. Rossi, M. Abderrahim, J. C. D´ıaz, Tracking moving optima using kalmanbased predictions, Evolutionary Computation 16 (1) (2008) 1–30.
[59] A. Simo˜es, E. Costa, Improving prediction in evolutionary algorithms for dynamic environments, in: . Raidl et al (Ed.), Genetic and Evolutionary Computation Conference, GECCO, ACM, Montreal, Qu´ebec, Canada, 2009, pp. 875–882.
[60] P. A. N. Bosman, H. L. Poutr´e, Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, New York, NY, USA, 2007, pp. 1165–1172.
[61] J. Branke, D. Mattfeld, Anticipation and flexibility in dynamic scheduling, International Journal of Production Research 43 (15) (2005) 3103–3129.
[62] R. K. Ursem, Multinational GA optimization techniques in dynamic environments, in: D.Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, H.-G. Beyer (Eds.), Genetic and Evolutionary Computation Conference, GECCO, Morgan Kaufmann, 2000, pp. 19–26.
[63] S. Yang, Constructing dynamic test environments for genetic algorithms based on problem difficulty, in: IEEE Congress on Evolutionary Compu1407 CEC, Vol. 2, 2004, pp. 1262–1269.
[64] P. A. N. Bosman, Learning, anticipation and time-deception in evolutionary online dynamic optimization, in: S. Yang, J. Branke (Eds.), GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization, 2005.
[65] T. T. Nguyen, Z. Yang, S. Bonsall, Dynamic time-linkage problems - the challenges, in: IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, 2012, in Press.
[66] T. T. Nguyen, X. Yao, Continuous dynamic constrained optimization- the challenges, IEEE Transactions on Evolutionary Computation (In Press)[online].
URL\url{http://www.staff.ljmu.ac.uk/enrtngu1/Papers/Nguyen_Yao_DCOP.pdf}
[67] T. T. Nguyen, X. Yao, Benchmarking and solving dynamic constrained problem in: IEEE Congress on Evolutionary Computation, CEC, IEEE Press, 2009, pp. 690–697.
[68] H. Richter, Memory design for constrained dynamic optimization problems, in: The European Conference on the Applications of Evolutionary Computation, Evo Applications, Vol. 6024 of Lecture Notes in Computer Science, Springer, 2010, pp. 552–561.
[69] H. G. Cobb, J. J. Grefenstette, Genetic algorithms for tracking changing environments, in: International Conference on Genetic Algorithms, Morgan Kaufmann, 1993, pp. 523–530.
[70] K. Trojanowski, Z. Michalewicz, Searching for optima 1430 in non-stationary environments, in: IEEE Congress on Evolutionary Computation, CEC, Vol. 3, IEEE, 1999, pp. 1843–1850.
[71] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, P. . N. Suganthan, Benchmark Generator for CEC 2009 Competition on Dynamic Optimization, Tech. rep., University of Leicester and University of Birmingham, UK (2008).
[72] M. Abello, L. T. Bui, Z. Michalewicz, An adaptive approach for solving dynamic scheduling with time-varying number of tasks - part I, in: IEEE Congress on Evolutionary Computation, CEC, 2011, pp. 1711 – 1718.
[73] Adaptive encoding for aerodynamic shape optimization using evolution strategies, Vol. 1.
[74] Y. Jin, M. Olhofer, B. Sendhoff, On evolutionary optimization of large problems using small populations, in: The First International Conference on Advances in Natural Computation, ICNC 2005, 2005, Part II, 2005, pp. 1145–1154.
[75] Y. Jin, B. Sendhoff, Constructing dynamic optimization test problems using the multi-objective optimization concept, in: G. R. Raidl (Ed.), Applications of evolutionary computing, Vol. 3005 of Lecture Notes in Computer Science, Springer, 2004, pp. 525–536.
[76] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: test cases, approximations, and applications, IEEE Transactions on Evolutionary Computation 8 (5) (2004) 425–442.
[77] M. Helbig, A. Engelbrecht, Archive management for dynamic multi-objective optimization problems using vector evaluated particle swarm optimization, in: IEEE Congress on Evolutionary Computation, CEC, 2011, pp. 2047 – 2054.
[78] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Prediction-based pop1ulation re-initialization for evolutionary dynamic multiobjective optimization, in: S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, T. Murata (Eds.), Evolutionary Multi-Criterion Optimization, Vol. 4403 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 832–846.
[79] X. Yu, Y. Jin, K. Tang, X. Yao, Robust optimization over time – a new perspective on dynamic optimization problems, in: IEEE Congress on Evolutionary Computation, CEC, Spain, 2010, pp. 3998–4003.
[80] R. W. Morrison, K. A. DeJong, A test problem generator 1465 for non-stationary environments, in: IEEE Congress on Evolutionary Computation, CEC, Vol. 3, IEEE, 1999, pp. 2047–2053.

[81] R. Morrison, Performance measurement in dynamic environments, in: J. Branke (Ed.), GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2003, pp. 5–8.
[82] J. J. Grefenstette, Evolvability in dynamic fitness landscapes: A genetic algorithm approach, in: IEEE Congress on Evolutionary Computation, CEC, Vol. 3, IEEE, 1999, pp. 2031–2038.
[83] K. Weicker, N. Weicker, Dynamic rotation and partial visibility, in: IEEE Congress on Evolutionary Computation, CEC, 2000, pp. 1125–1131.
[84] W. Tfaili, J. Dr´eo, P. Siarry, Fitting of an ant colony approach to dynamic optimization through a new set of test functions, International Journal of Computational Intelligence Research 3 (2007) 205–218.
[85] R. Tinos, S. Yang, Continuous dynamic problem generators for evolutionary algorithms, in: IEEE Congress on Evolutionary Computation, CEC, 2007, pp. 236–243.
[86] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Tech. rep., Nanyang Technology University, Singapore (2005).