Do We Need Change Detection for Dynamic Optimization Problems?

Abdennour Boulesnane[1] and Souham Meshoul[2]

[1] Faculty of Medicine, Salah Boubenider-Constantine 3 University, Constantine, Algeria
[2] Princess Nourah Bint Abdulrahman University RC-CCIS, Riyadh, Saudi Arabia
aboulesnane@univ-constantine3.dz
sbmeshoul@pnu.edu.sa

Abstract. Solving dynamic optimization problems is more challenging than static ones. When a change in the objective landscape occurs, the search process may not be powerful enough to track new optima. For population based algorithms this is referred to as diversity loss problem. Furthermore, the memory of old optima becomes outdated and if not correctly dealt with, the evolution of the search process may be misguided. Recently, a new interesting trend in dealing with optimization in dynamic environments has emerged toward developing new algorithms that are able to effectively handle changes without using any change detection scheme, and hence no extra computational cost is needed. There exist several works in the literature that attempt to maintain diversity without change detection. However, not that much work has been devoted to studies that investigate the possibility to overcome the outdated memory problem without expensive change detection. This study presents a comprehensive survey of the various change detection based methods. As part of this survey, we include a classification of the change detection schemes and we identify the main features of each method.

Keywords: Dynamic optimization problems, Change detection, Diversity loss problem, Outdated memory problem, Memory schemes

1 Introduction

Many real world problems require optimization over time because of the dynamic nature of the environments. Typical fields where such problems need to be solved include economics, engineering, communication systems, machine learning, bioinformatics to name just a few. Time dependent optimization problems are most commonly known as dynamic optimization problems (DOPs). Solving DOPs is not only a matter of locating global optima as in static optimization but of being able to track such optima in changing objective landscapes as well. Hence, a DOP can be viewed as a sequence of static optimization problems over time [28]. DOPs have been defined in different ways. A unified description can be found in [9]. Over the past decade, Swarm Intelligence (SI) [18] and Evolutionary Algorithms (EAs) [23] are considered to be a good choice for solving DOPs.

However, two main problems are encountered when using traditional methods to solve DOPs. The first one is the diversity loss caused by the convergence of these approaches preventing them from tracking new optima in an efficient manner. The second one is the outdated memory caused by changes in the environment and that may misguide the evolution of the search process.

On the other hand, dynamic change detection is an integral part of dynamic evolutionary algorithms design. Following the change detection, the diversity loss and the outdated memory problems can be efficiently resolved by reacting properly to the environmental change using mechanism such as [33]: reevaluation of population to update memory, clearing the memory, introducing diversity, re-initialization of parameters of the algorithm, etc. For that purpose, different kinds of change detection schemes have been proposed in the literature [25]. However, difficulties in detection schemes include proper choice of memory solutions that represent the whole search space, increased computational evaluation cost, detecting partial change in the landscape, noisy environments, etc.

To avoid these drawbacks, recently, a new interesting trend in dealing with dynamic environments has emerged toward developing new algorithms that are able to effectively handle dynamics without any change detection schemes by focusing on the optimization process rather than spending computational resources on change detection. Therefore, several studies in the literature have been carried out in an attempt to maintain diversity without change detection [15]. On the other hand, very little work has been done to investigate the possibility to overcome the outdated memory problem without expensive change detection.

In this paper, we analyze the existing state of the art change detection based methods proposed for the dynamic environments in the literature. Moreover, we present for first time a classification of these schemes and highlight their advantages and limitations. Furthermore, we discuss the importance of strategies that do not require the knowledge of the change point time to handle future changes and what kind of factors must be taken into consideration to move towards this new dynamic optimization design framework.

The rest of the paper is organized as follows. Firstly, problem statement and challenges are presented in section 2. Section 3 and 4 describe change detection and without change detection based methods respectively. Finally, conclusions and plans for future work are given in section 5.

2 Dynamic optimization problems (DOPs): Problem statement and Challenges

Optimization in dynamic environments, popularly known as *dynamic optimization*, was, and still is, an active open research area, due to its obvious and direct relation to real-world problems. The main challenge in the field of optimization is the dynamic nature of these problems which imposes to find solutions subject to time constraints. We can formally describe a dynamic optimization problem

as the task that aims to find the sequence $(x_1^*, x_2^*, ..., x_n^*)$ that:

$$\begin{array}{l} \text{Optimize } f(x,t) \\ \text{subject to. } h_j(x,t) = 0 \text{ for } j = 1, 2, ..., u \\ g_k(x,t) \leq 0 \text{ for } k = 1, 2, ..., v. \\ \text{with } x \in \mathbb{R}^n \end{array} \quad (1)$$

Where $f(x,t)$ is a time dependent objective function, $(x_1^*, x_2^*, ..., x_n^*)$ is the sequence of n optima found as the fitness landscape changes. In other ways, it depicts optima tracking, $h_j(x,t)$ denotes the j^{th} equality constraint and $g_k(x,t)$ denotes the k^{th} inequality constraint. All these functions may change dynamically at any time, as shown by the dependence on the time parameter t. When applied to DOPs, the objective of an optimization algorithm is no longer to simply find the optimal solution, but also to track its movement in a dynamic search space. Hence, traditional optimizers have been revisited and significant improvements have been introduced to ensure appropriate behavior when dealing with dynamic environments [33].

2.1 Diversity loss problem

One of the most important challenge posed by the dynamic behavior of DOPs is the diversity loss issue [33]. The diversity loss occurs when all solution candidates converge to the same region in the search space and lose their ability to find new optimum as required after any change in the environment. In the literature, several approaches have been developed to address this limitation. Indeed, at the origin most of them were initially designed to handle static optimization problems, including approaches based on swarm intelligence and evolutionary algorithms. To fit the purpose, they have been intelligently adapted to cope with dynamic environments by using some strategies such as [33]: maintaining diversity during the run, increasing diversity after a change, using several populations (multi-population approaches), using memory schemes, using prediction mechanisms and hybrid approaches.

Memory schemes to tackle diversity loss is a connecting link between all these strategies, where almost in every study at least one memory-based method was used. Despite the large variety of memory schemes, they all rely on a core and common feature which is the reuse of information accumulated during the optimization process to enhance their adaptability to the environmental change. According to a comprehensive study on various memory schemes [4], the most popular memory approach is the use of an explicit memory to store the good solutions collected over the successive environments, known as *global memory scheme*.

2.2 Outdated memory problem

Whatever the adopted memory pattern is (explicit, implicit, global, local, associative, direct, etc.), the issue that makes the dynamic optimization task more

challenging is the outdated memory problem. This problem refers to the condition in which all existing information that the dynamic optimization algorithm has accumulated during the search process (i.e. stored personal best and/or global best positions and/or their corresponding function values, etc.) may no longer be useful or even valid after a dynamic change. Therefore, incorporating directly this stored knowledge into the search process has the potential to negatively affect and mislead the dynamic optimizer in its quest to follow the global optimum in the changing environment. In the literature, the outdated memory problem has been solved in two ways:

- Using change detection approaches by either re-evaluating or forgetting the memory when changes in the environment occur.
- Without change detection by iteratively re-evaluating the memory contents [14].

Although the efficiency of using change detection approaches or iterative function re-evaluation to overcome the outdated memory problem was well proved, the optimization task becomes time and effort intensive. Therefore, a new research direction has been followed [14] which consists of designing new dynamic optimization algorithms able to cope with DOPs adaptively without using any change detection process while saving a large amount of computational resources. In this context and motivated by these facts, the goal of this paper is to refer to the need to think not only of the diversity problem, but also of solving the outdated memory problem by using new methods with the aim of laying a solid foundation for a new dynamic optimization design framework.

The aforementioned issues have been tackled in different ways in the literature. Related methods can be broadly classified into different categories according to whether they make use of a change detection mechanism or not.

In the rest of this section, we present a review of methods in both classes.

3 Change detection based methods

For solving the change detection task, a change point $t_{cp} \in \mathbb{N}_0$ can be defined as follows [25]:

$$f(x, t_{cp}) \neq f(x, t_{cp} + 1) \tag{2}$$

where $x \in M$ is an element of a fixed bounded search space $M \subset \mathbb{R}^n$. The change point definition in Eq. (2) says that a change in the function landscape has taken place no matter how small and irrelevant the alteration in the environment is. Besides, approaches based on change detection need to know this moment to react properly to the dynamic environments. Hence, a change detection mechanism should be implemented.

Once the change is detected, the algorithm takes explicit mechanism (change reaction schemes) to respond to changes and to increase or introduce diversity in population, therefore, tracking moving optima becomes easy. The following mechanisms are widely utilized : a) employing memory [26]; b) hyper mutating

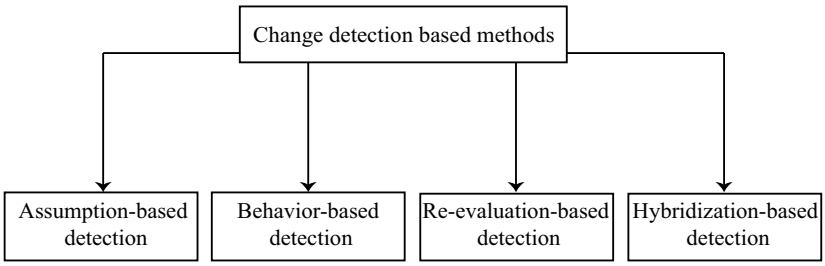

Fig. 1: Different schemes in change detection based methods.

the previous population [19]; c) randomly generating new solutions [31]; and d) anticipating and predicting [26], etc.

In this section, and as shown in Fig. 1, we present a classification and an overview on the state of the art of change detection policies that have been used to deal with dynamic environment.

3.1 Assumption-based detection

Many dynamic optimization approaches take an explicit action to detect changes in the environment. The simplest way to detect the environmental changes is to assume that changes in the environment are either known a priori to the algorithm or can be easily predicted. Assumption based approaches do not spend any additional function evaluations to detect changes. Usually, in the state-of-the-art experimental studies, this method allows to fairly evaluate and analyze the performance of the proposed techniques offering an ideal situation where all changes are 100% detected [29]. Furthermore, this method can be used to handle with periodic DOPs [30] in which the changes are deterministic and predictable, hence the system can easily calculate the time of the next change.

This method makes sense for solving the academic benchmark problems. However, it may be inefficient, especially in cases where the changes are often random or unpredictable (i.e. real-world applications).

3.2 Behavior-based detection

Changes that influence the search space, can influence the algorithm's behavior as well. Based on this idea, behavior based approaches try to exploit the irregularities in the algorithm behavior in order to confirm the occurrence of changes in the environment. A change detection step is needed when the algorithm has no earlier information of the time of next change. In many studies, monitoring the quality of the best solutions in the population can be used as a change detection mechanism. Either by observing the drop in the average of best stored solutions during a certain number of iterations [22] or if these archived solutions do not change over a number of generations [12], then it is inferred that the change has taken place.

[10] proposed Partitioned Hierarchical Particle Swarm Optimizer (PH-PSO) for dynamic and noisy function optimization. PH-PSO divides the swarm into a tree-based hierarchy of sub-swarms for a certain number of iterations after the occurrence of a change. A new approach to detect environmental changes under the presence of noise is introduced, called hierarchy monitoring change detection. In this method, change detection is performed by observing changes in the hierarchy itself. Doing so does not involve any additional evaluation of the fitness function and allows working in noisy environment.

Another interesting idea has been investigated in [25] in order to detect changes in a systematic way by analyzing the behavior of population over generations. The key idea is to find difference between fitness distributions of the populations through non-parametric statistical hypothesis tests. Performance degradation over consecutives generations is taken as an indicator of change detection. However, the used non-parametric tests involve using independent samples which is not satisfied given fitness distributions [27]. To address this issue, an immunological approach using negative selection has been proposed in [24].

In [20], a detection mechanism is proposed by which the deviation amount of the locally mutated Differential Evolution (DE) individuals is tracked to detect changes. The basic idea relies on the fact that locally mutated DE offsprings differ slightly from their parents resulting in small fitness difference. However, the deviation amount is expected to be significant with dynamic change because offspring are produced on a thoroughly revised landscape whereas parents keep the fitness obtained in the previous generation. Hence, tracking such deviation would be good change indicator. As a consequence, neither extra functional evaluations are involved nor high complexity computations.

In [7], a change detection operator is introduced base on the objective function values of the individuals at successive iteration. A value if this operator greater than a predefined value can be interpreted as change occurrence.

By using this schemes, there is no need for additional function evaluations. However, since behavior based approaches are mainly related to the algorithms used, they can be defined as specific approaches, which means it is impossible to directly incorporate these approaches into other algorithms.

3.3 Reevaluation-based detection

Re-evaluating solutions is the most commonly used change detection approach [23]. Some specific solutions in the search space (named detectors or sensors) are employed and then their objective values are evaluated after each iteration. If the present and past objective values are different, then, indeed, the environment has changed. In the literature, numerous methods have been developed with the integration of this change detection scheme. They differ by each other by the strategy they use to place sensors in the landscape, to determine the required number of sensors and the kind of solutions that these sensors should represent. Accordingly, the type of the used detectors influences the performance of the detection. Solution used for change detection can be part of the population as

in [11], where a percentage of solution from the current population selected randomly is re-evaluated. Otherwise, in [13], change detection is done using a random point called test solution. The fitness value of this test solution is calculated at every generation. If a significant difference is recorded, this can be an indication of a change in the environment. Also in [6], the change detection problem is addressed by using a fixed set of detectors that are not part of the swarm and distributed in the search space in a way to cover its parts more evenly.

On the other hand, stored points from the memory can be used as change detectors as in [5] where some detectors are re-evaluated to state whether a change has taken place or not. For instance in [16], the top 5 best solutions at each iteration are considered in the re-evaluation process. In [21], the current best solution is taken as a detector. This may cover only some part of the landscape. Therefore, a multi-population approach has the advantage to cover more parts of the landscape. Also in [17], where cooperative methods for DOPs that are based on a set of cooperating agents are used. an agent re-evaluates its best performance recorded so far and checks the extent to which it changes.

Furthermore, in [1], an empirical study has been conducted to thoroughly study the impact of how to distribute a set of detectors in the search space. The performance of four different sensor-based detection schemes has been investigated, which are: random selection of individuals from the population, best-so-far individuals, randomly selected individuals that do not belong to the population and distributing sensors. Each placement strategy has been measured by using two commonly known dynamic optimization problems.

It is worth noting that this approach is easy to implement, as it does not require elaborated statistical analysis and it is more favorable when the change detection is more difficult [25]. However, it assumes that there is no noise in function evaluations, which means it is not robust with all problems instances. Moreover, it spends a large amount of computation throughout the search space on detecting the changes.

3.4 Hybridization-based detection

All existing change detection methods have limitations, an efficient way to overcome these drawbacks is the hybridization between two or more change detection approaches. As in [2], a hybrid change detection schemes for dynamic optimization problems is proposed. In this study, the well known statistical hypothesis testing approaches (Behavior-based detection) are combined with three sensor-based detection schemes (Reevaluation-based detection) in order to increase detection capability.

Despite the effectiveness of the hybrid change detection scheme, it is always suffering from the problem of high computational costs.

4 Without change detection based methods

Some dynamic environments can pose challenges such as noise or partial landscape changes, which will be hard or even impossible to properly detect change

in. Therefore, dynamic optimization algorithms in such environments depend for their success on an efficient change detection method and an adequate change reaction strategy. However, and as has been explained above, the change detection schemes in the literature are also suffering from several performance issues especially the high computational costs.

To avoid these drawbacks, a new initiative has been recently undertaken in [14]. This new idea consists of developing new algorithms that will be able to deal with dynamic environments adaptively without using any change detection schemes. In other words, solving DOPs without using any change detection methods to deal with the diversity loss problem and the outdated memory problem.

In fact, there has been much literature on the subject of the diversity loss problem. Generally, this problem has been tackled with the ultimate goal to maintain population diversity without using any change detection mechanism throughout the entire run of the algorithm [32]. However, these methods show the convergence and focusing constantly on diversity may affect the optimization process in an undesirable way. Multipopulation approaches have been proposed as an effective way to address the problem of diversity loss [14] [15]. For example in [14], when the population diversity drops below a certain level, a strategy that selects random immigrants is performed to inject them into the population and boost its diversity.

On the other hand, the outdated memory problem has been always solved using iterative re-evaluation which is computationally expensive. This problem may prevent the algorithm finding new optima and misguide the optimization process. Therefore, solving this issue is of crucial importance. However, very little research effort has been devoted to it compared to diversity loss problem solving. One example is the evaporation mechanism proposed in [8]. This mechanism proposes to reduce the fitness value of the best position found by each particle during the search process. Therefore, better performance can be achieved by spending the efforts avoided in re-evaluation stored solutions, on the optimization process.

In the same context, a recent new Q-learning RL model has been proposed to solve DOPs [3]. This RL model does not require any change detector to know the occurrence of future changes in the environment. Rather, it is the task of the learner agent to learn how to deal with different complex situations.

5 Conclusion

Solving DOPs without using any change detection scheme is an open research issue. Whatever the problem's characteristics and whenever the changes occur, this kind of algorithms is not based on the knowledge of the change point time to handle future changes. Therefore, they will be effective for solving hard problems with recurrent changes and fast changes due to the fact that these algorithms do not need to spend any unnecessary effort on detecting changes.

In this paper we have tried to provide a survey of the literature on various change detection based schemes. We have also proposed a new classification of change detection based methods.

For future work, it would be interesting to design a new optimization algorithm that can dynamically adapt to changes by maintaining diversity and remedy the problem of outdated information without change detection.

References

1. Altin, L., Topcuoglu, H.R.: Impact of sensor-based change detection schemes on the performance of evolutionary dynamic optimization techniques. Soft Comput **22**, 4741–4762 (2018)
2. Altin, L., Topcuoglu, H.R., Ermis, M.: Hybridizing change detection schemes for dynamic optimization problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2086–2093. San Sebastian (2017)
3. Boulesnane, A., Meshoul, S.: Reinforcement learning for dynamic optimization problems. In: Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion). ACM, Lille, France (2021)
4. Bravo, Y., Lugue, G., Alba, E.: Global memory schemes for dynamic optimization. Natural Comput **15**, 319–333 (2016)
5. Bu, C., Luo, W., Yue, L.: Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies. IEEE Trans. Evol. Comput **21**, 14–33 (2017)
6. Campos, M., Krohling, R.A.: Entropy-based bare bones particle swarm for dynamic constrained optimization. Knowledge-Based Systems **97**, 203–223 (2016)
7. Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Trans. Evol. Comput **8**, 425–442 (2004)
8. Fernandez-Marquez, J.L., Arcos, J.L.: An evaporation mechanism for dynamic and noisy multimodal optimization. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 17–24. Montreal, Québec, Canada (2009)
9. Fu, H., Lewis, P.R., Sendhoff, B., Tang, K., Yao, X.: What are dynamic optimization problems? In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1550–1557. Beijing (2014)
10. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer for noisy and dynamic environments. Genet Program Evolvable Mach **7**, 329–354 (2006)
11. Jiang, S., Yang, S.: A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization. IEEE Trans. Evol. Comput **21**, 65–82 (2017)
12. Jordehi, A.R.: Particle swarm optimisation for dynamic optimisation problems: a review. Neural Comput and Applic **25**, 1507–1516 (2014)
13. Kundu, S., Biswas, S., Das, S., Suganthan, P.N.: Crowding-based local differential evolution with speciation-based memory archive for dynamic multimodal optimization. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 33–40. Amsterdam, The Netherlands (2013)
14. Li, C., Yang, S.: A general framework of multipopulation methods with clustering in undetectable dynamic environments. IEEE Trans. Evol. Comput **16**, 556–577 (2012)

15. Li, C., Yang, S., Yang, M.: An adaptive multi-swarm optimizer for dynamic optimization problems. Evolutionary computation **22**, 559–594 (2014)
16. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 51–58. Seattle, Washington, USA (2006)
17. Masegosa, A.D., Pelta, D., Amo, I.G.D.: The role of cardinality and neighborhood sampling strategy in agent-based cooperative strategies for dynamic optimization problems. Appl Soft Comput **14**, 577–593 (2014)
18. Mavrovouniotis, M., Li, C., Yang, S.: A survey of swarm intelligence for dynamic optimization: Algorithms and applications. Swarm Evol. Comput **33**, 1–17 (2017)
19. Morrison, R.W., Jong, K.A.D.: Triggered hypermutation revisited. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), pp. 1025–1032 vol.2. La Jolla, CA (2000)
20. Mukherjee, R., Debchoudhury, S., Das, S.: Modified differential evolution with locality induced genetic operators for dynamic optimization. Eur J Oper Res **253**, 337–355 (2016)
21. Mukherjee, R., Patra, G.R., Kundu, R., Das, S.: Cluster-based differential evolution with crowding archive for niching in dynamic environments. Inf Sci (Ny) **267**, 58–82 (2014)
22. Nguyen, T.T.: Continuous dynamic optimisation using evolutionary algorithms. Ph.D. thesis, University of Birmingham, Birmingham, U.K (2011). URL http://etheses.bham.ac.uk/1296
23. Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: A survey of the state of the art. Swarm Evol Comput **6**, 1–24 (2012)
24. Richter, H.: Change detection in dynamic fitness landscapes: An immunological approach. In: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), pp. 719–724. Coimbatore (2009)
25. Richter, H.: Detecting change in dynamic fitness landscapes. In: 2009 IEEE Congress on Evolutionary Computation, pp. 1613–1620. Trondheim (2009)
26. Richter, H., Yang, S.: Learning behavior in abstract memory schemes for dynamic optimization problems. Soft Comput **13**, 1163–1173 (2009)
27. Richter, H., Yang, S.: Dynamic optimization using analytic and evolutionary approaches: A comparative review. In: I. Zelinka, V. Snášel, A. Abraham (eds.) Handbook of Optimization, p. 1–28. Springer-Verlag, Berlin, Germany (2013)
28. Rohlfshagen, P., Yao, X.: Dynamic combinatorial optimisation problems: an analysis of the subset sum problem. Soft Comput **15**, 1723–1734 (2011)
29. Sahmoud, S., Topcuoglu, H.R.: A memory-based nsga-ii algorithm for dynamic multi-objective optimization problems. In: European Conference on the Applications of Evolutionary Computation, pp. 296–310 vol. 9598. Porto, Portugal (2016)
30. Tinós, R., Yang, S.: Analyzing evolutionary algorithms for dynamic optimization problems based on the dynamical systems approach. In: S. Yang, X. Yao (eds.) Evolutionary computation for dynamic optimization problems, pp. 241–267. Springer-Verlag, Berlin, Germany (2013)
31. Tinós, R., Yang, S.: A self-organizing random immigrants genetic algorithm for dynamic optimization problems. Genet Program Evolvable Mach **8**, 255–286 (2007)
32. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput **9**, 815–834 (2005)
33. Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y., Yao, X.: A survey of evolutionary continuous dynamic optimization over two decades – part a. IEEE Transactions on Evolutionary Computation pp. 1–1 (2021)