Self-adaptation in dynamic environments – a survey and open issues

Pavel Novoa-Hernández*
Department of Mathematics,
University of Holguin,
Holguin, Cuba
Email: pnovoa@facinf.uho.edu.cu
*Corresponding author

Carlos Cruz Corona and David A. Pelta
Department of Computer Science and Artificial Intelligence,
Center for Research in Information and Communication Technologies (CITIC-UGR),
University of Granada,
Granada, Spain
Email: carloscruz@decsai.ugr.es
Email: dpelta@decsai.ugr.es

Abstract: Self-adaptation is a popular parameter control technique in evolutionary computation, which has been extensively studied in stationary optimisation. In the context of dynamic optimisation problems (DOPs), there are research works that suggest the application of such technique. Nevertheless, some important issues remain open, for example, how self-adaptation can be more profitable for a given algorithm. From the survey we made, it is possible to distinguish three main application levels of self-adaptation in dynamic environments: metaheuristic level, ‘mechanism for DOPs’ level, and the combination of both. While most of the related works belong to the first level, a small number can be grouped in the second one. However, in contrast to previous two, unfortunately, very little or nothing has been done with the third one. Based on these motivations, in this paper we empirically analysed the role of several self-adaptive models in these levels using multipopulation differential evolution algorithms as baseline. The results suggest that self-adaptation has a significant impact when applied at least to the ‘mechanism for DOPs’ level.

Keywords: self-adaptation; dynamic environments; differential evolution; bio-inspired algorithms.


Biographical notes: Pavel Novoa-Hernández is an Associate Professor at University of Holguin, Cuba. In 2007, he obtained his Bachelor in Computer Science Engineering at University of Holguin, and Master of Science in Computer Science and Artificial Intelligence by the Central University of Las Villas, Cuba in 2010. He received his PhD in Computer Science by the University of Granada, Spain in 2013. His research interests are related to evolutionary computation, dynamic optimisation, applied mathematics and software development. Currently, he is a member of the Cuban Society of Mathematics and Computation.

Carlos Cruz Corona received his MS in Electronic Engineering in 1986 from the Central University of Las Villas, Cuba; and PhD in Computer Sciences from the University of Granada, Granada, Spain, in 2005. He is a Lecturer at Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Spain. He has published several scientific and technical papers in leading scientific journals. He has served on many international programme committees and has attended numerous national and international conferences, congresses, and workshops. He has participated in a variety of national and international research and educational projects. His current scientific interests are soft computing, fuzzy sets and systems, decision support systems, metaheuristic algorithms, and nature inspired systems.

David A. Pelta is an Associate Professor at the Department of Computer Science and AI, University of Granada, Spain. He obtained his Computer Science degree in 1998 from the National University of La Plata, Argentina and PhD in 2002 from the University of Granada, Spain. He is a member of the Models of Decision and Optimisation Research Group, where he conducts research on soft computing techniques, cooperative strategies for optimisation,
adversarial reasoning and interdisciplinary applications. He is actively involved in research projects funded by different organisations (Spanish Government, European Community, Andalusian Government, etc.). He has published more than 35 journal papers, co-edited seven books and five special issues on relevant journals. He co-advised six PhD and four Master’s thesis. More details are available at his webpage http://decsai.ugr.es/dpelta.

1 Introduction

There are several real-world scenarios that can be modelled as dynamic optimisation problems (DOPs). These are characterised by the dynamic nature of the model elements, for example: the objective function, the search space, etc. Particularly, in the present work we consider DOPs with time-varying objective function and search space. Formally, \( \Omega^t \subseteq \mathbb{R} \) the search space at time \( t \), \( (t \in \mathbb{N}_0) \), a DOP is defined as the systematic search of the set of optimal solutions \( X^t \) such as:

\[
X^t = \{ x_{opt}^t : f^t(x_{opt}^t) \leq f^t(x), \forall x \in \Omega^t \}
\]

where \( f^t : \Omega^t \rightarrow \mathbb{R} \) is the objective function at time \( t \).

Solving DOPs by metaheuristics has gained a great interest in the last years (Jin and Branke, 2005; Cruz et al., 2011; Nguyen et al., 2012; Pelta et al., 2009; Novoa-Hernández et al., 2010, 2011) because their suitability to deal with complex scenarios, and the incorporation of specific mechanisms to face the dynamics of the problem. In this later case, according to Jin and Branke (2005), the current mechanisms for DOPs can be roughly grouped as follows: diversity during the run, diversity after the changes, memory approaches, and multipopulation approaches.

Recently, as Nguyen et al. (2012) pointed out: “another approach is to make use of the self-adaptive mechanisms of (...) meta-heuristics to cope with changes”. Self-adaptation is a commonly known parameter control technique in evolutionary computation, and has been extensively studied in stationary environments (Angeline, 1995; Bäck, 1997; Eiben et al., 2007; Meyer-Nieberg and Beyer, 2007). Its main characteristic is the intelligent control of the algorithm behaviour during the run.

Since the changes in DOPs affect the algorithm behaviour over time, it is reasonably to expect that the application of self-adaptation in this context improves the optimisation process. Some works support this point of view (e.g., Angeline et al., 1996; Bounanza, 2005; Brest et al., 2009; du Plessis and Engelbrecht, 2011; Novoa-Hernández et al., 2013), but there are still some open issues related to this topic that need to be researched. Among them, one can find the one regarding how and where to apply self-adaptation in an algorithm. It is well-known that some classical metaheuristics have certain inherent self-adaptive behaviour (e.g., evolution strategy, real-coded genetic algorithms – GAS). However, according to studies such as Angeline (1995), and Weicker and Weicker (1999), such a behaviour could be not enough to deal with problems under dynamic environments. Therefore, an alternative would be to apply self-adaptation in mechanisms or extensions designed to deal with the dynamics of the problem.

Recently, Novoa-Hernández et al. (2013) showed how to improve the performance of a metaheuristic applying a self-adaptive strategy to generate diversity throughout the run. As the reader might notice, another option would be to have self-adaptation at both levels, metaheuristic and mechanism designed to deal with DOPs. As far as we know, this idea has not been explored yet.

Based on the above motivations, the general goal of this paper is to review the role of self-adaptation in dynamic environments. Specifically, we have studied the impact of certain self-adaptive strategies, in combination (or not) with a self-adaptive approach for DOPs (diversity during the run). To achieve this, we have focused our analysis in the context of multipopulation differential evolution (DE) algorithms (Storn and Price, 1997). The DE metaheuristic has proven to be an efficient method for continuous domains, which is the case of the DOPs considered in this work. Our hypothesis is that self-adaptation can be more effective when it is incorporated (at least) into a mechanism for DOPs.

The rest of this paper is organised as follows: Section 2 provides a survey on self-adaptation in dynamic environments, in which current research has been organised so that the reader can easily identify the open issues mentioned above, and some research opportunities. In that sense, an experimental study is conducted in Section 3 in order to empirically analyse these open issues. Here, it is also proposed an intuitive way to analyse the adaptability level of self-adaptive algorithms in dynamic environments. Finally, Section 4 is devoted to the conclusion and future works.

2 Self-adaptation in dynamic environments

As it was mentioned before, self-adaptation is, from the point of view of evolutionary computation, an effective parameters control technique. Essentially, according to Meyer-Nieberg and Beyer (2007):

“a self-adaptive algorithm controls the transmission function between parent and offspring population by itself without any external control. Thus, the concept can be extended to include algorithms where the representation of an individual is augmented with genetic information that does not code information regarding the fitness but influences the transmission function instead.”
Although including strategy parameters at the individual representation is the common way to ensure a self-adaptive behaviour in the algorithm, works like Beyer and Deb (2001), and Igel and Toussaint (2003) showed that it is possible to observe such a behaviour implicit in the algorithm (e.g., in some crossover operators of real-coded GAs).

In what follows, we survey the most important works on self-adaptation in dynamic environments. The search was conducted using the Scopus\textsuperscript{1} engine and the query TITLE(self-ad* AND (dynam* OR non-stationary OR time-vari*) AND (optim* OR environment OR problem OR model)). Besides, we traced relevant references from certain papers. A human-made selection resulted in 27 papers from conferences and journals.

Perhaps the first works on self-adaptation in dynamic environments were Angeline’s studies (Angeline et al., 1996; Angeline, 1997). In the first one, the author focused in the comparison of two self-adaptive update rules: the lognormal mutation (Schwefel, 1981) and a Gaussian variant (Fogel et al., 1991). As baseline algorithm, the authors selected a multi-mutational self-adaptive evolutionary programme consisting in five modes of mutations for the individuals. Angeline used a rudimentary prediction task over three different scenarios as the test bed. The main conclusion of this research was the success of the self-adaptive variants over the non-adaptive counterparts. In particular, the Gaussian mutation scheme was found slightly advantageous over the lognormal variant.

In the second work, Angeline (1997) studied the performance of two evolutionary algorithms (EAs), one with a Gaussian self-adaptation model, while the other with an adaptive update rule based in the individual fitness. Angeline concluded that for almost all tested scenarios the self-adaptive variant experimented a chaotic behaviour: more accurate at the beginning, but losing the tracking ability during the run. On the other hand, the adaptive variant was not only capable to achieve a better performance but also to experiment a fixed tracking pattern over time.

Other similar works on EAs were conducted by Bäck (1997, 1998, 1999), and Greffenstette (1999) in which GAs includes self-adaptive mutation rates. Specifically, these GAs used self-adaptation in what the authors called the hypermutation rate, which can be seen as a case of self-adaptation at the level of the mechanism for DOPs.

Similar to Angeline (1997), Weicker and Weicker (1999) carried out an interesting comparative analysis among four evolution strategies algorithms with different mutation mechanisms: uniform step-adaptation (only one strategy parameter), separate step-adaptation (several strategy parameters), covariance matrix adaptation (CMA) (Hansen and Ostermeier, 1996), and spherical step-adaptation. The last one was proposed in order to solve the problems presented by sophisticated mutation mechanisms, such as the separate step-adaptation and the CMA. Using as a test bed a dynamical spiral function, they found that the most sophisticated approaches were the worst behaved. In the case of the CMA, this mechanism improves its performance as the number of the offspring increases (e.g., greater than 50). Surprisingly, simple mutation mechanisms such as the uniform step-adaptation mechanism and the spherical one, showed a stable behaviour in the considered scenarios.

In Deb and Beyer (2001), the self-adaptive behaviour of a real-parameter GA with simulated binary crossover (SBX) and the self-adaptive evolution strategy (SA-ES) were compared. Among the scenarios used for the experiments, the authors studied for illustrative purposes a dynamic version of the sphere model with 30 dimensions. The dynamic of the changes was rather simple (i.e., varying the optima in the range $[-1, 1]$ every 1,000 generations). As results, the authors found that “GAs with SBX operator can quickly increase its population diversity and converge to the new optimum. First, the population gets more diverse to come near the current optimum and then decrease diversity to converge closer and closer to the current optimum”. Hence, a similar self-adaptive behaviour exists between the realparameter GAs with the SBX operator and SA-ES.

In Stanhope and Daida (1999), the authors theoretically analysed a $(1 + 1)$ mutation-only GA in a dynamic match fitness function. The main conclusions were that “small perturbations in the fitness function over time had large adverse effects on GA performance for a given mutation rate. Second, that for a given fitness function changing with the $(1, 1)$ dynamic, optimisation of the mutation rate resulted in little difference in terms of expected GA performance”. The authors explained that “this lack of performance improvement implies that techniques such as self-adaptation, which rely on an implicit optimisation of the mutation rate, will have little utility in this case.” In other words, they reported a negative result for self-adaptation in dynamic environments.

In 2002, an interesting theoretical analysis was conducted by Arnold and Beyer (2002). In this paper, a $(\mu/\mu, \lambda)$-ES with isotropic mutation (i.e., only one mutation strength) was analysed. The novelty of this research was the analysis of the cumulative mutation strength adaptation (Hansen and Ostermeier, 2001) in a sphere model with random dynamics. As a result, a scaling law depending on the distance to the problem optimum and the mutation strength was obtained. This expression was solved analytically yielding to an optimal mutation strength. From these results, the authors concluded that the cumulative mutation strength adaptation can ‘works perfectly’ because it is capable to generate the required (optimal) mutation step to approach the problem optima.

More recently, the same authors carried out a similar theoretical study on the sphere model with linear dynamics (Arnold and Beyer, 2006). This time, they concluded that although the ES mutation step-size is not optimal, the adaptation carried out by this self-adaptive model ensures a suitable tracking of the problem optima.

The learning capabilities of the mutation steps for both, the SA-ES and the covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier, 2001), were
studied by Boumaza (2005). As the test bed, the dynamic sphere was employed in combination with three types of optimum movements according to its velocity: constant, linear, and quadratic. The main conclusions were that both paradigms are capable to reflect the nature of the optimum movements in the self-adapted mutation steps. Besides, the CMA-ES can also learn the movement direction through the covariance matrix, that is, the dominant eigenvector has the same direction as the moving optimum.

Other experimental research using the SA-ES metaheuristic was carried out by Schönemann (2007). Here, the influence of the number of SA-ES mutation factors to guarantee a successful tracking is studied. The main conclusion of this work is that the number of mutation factors must be equals to the search space dimensions, even for problems with high severity of change.

A novel representation for GAs was proposed by Liang (2009). In addition to the decision variable vector, the author includes in the individual representation the dynamic fitness and the dynamic tendency. The resulting algorithm was tested on five artificial dynamic problems (designed by the author), and compared with other GAs. The results showed that the algorithm including this representation outperforms the others.

An interesting analysis on time series forecasting in non-stationary environments using adaptive and self-adaptive genetic programming (GP) algorithms, was conducted by Wagner and Michalewicz (2009). Regarding to the self-adaptive variant, the authors included at the individual representation, the so called window size parameter. That is, the size of the window of the training data series. However, although the proposal sounded interesting, so far there is no evidence that the authors have applied it in concrete problems.

A self-adaptive variant of the DE (Storn and Price, 1997) metaheuristic was presented in Brest et al. (2009), which they employed a multipopulation approaches. The employed method was the jDE extension proposed in Brest et al. (2006, 2007). Originally, jDE was designed for stationary optimisation, being its major characteristic the self-adaptation of the crossover rate (Cr) and the scaling factor (F). Besides employing several populations, this jDE extension also includes three novel mechanisms for DOPs:

1. an aging strategy to reinitialise individuals with certain age
2. a re-initialisation strategy for individuals that are close enough to the subpopulation best individual
3. the use of memories in such re-initialisation.

The method was tested on the problem instances defined by the CEC’2009 competition on dynamic environments (Li et al., 2008), which were derived from the generalised dynamic benchmark generator (GDBG) from Li and Yang (2008). It is important to highlight that this algorithm won this competition.

Similar to the previous work, du Plessis and Engelbrecht (2011) conducted a study on the effect of a novel multipopulation approach called competitive differential evolution, CDE, on three self-adaptive DE algorithms: SDE (Salman, 2007), jDE (Brest et al., 2006), and the Bare Bones DE (Omran et al., 2009). CDE evaluate the subpopulation sequentially according to their qualities, that is, the best one is continuously evolved up to its performance drops below that of another subpopulation. Besides, CDE also includes a novel crowding scheme, called re-initialisation midpoint check (RMC). The conducted experiments were performed on several instances of the moving peaks benchmark (MPB) (Branke, 1999) and the GDBG benchmark. The main conclusion of the work was regarding the benefits of the proposed multipopulation approach. Specifically, the derived algorithms outperformed some methods from the literature in certain scenarios. Although the aim of the authors was to study the influence of the proposed multipopulation approach, these works are interesting because it is a clear evidence that self-adaptation combined with clever approaches for DOPs, led to a more efficient optimiser in dynamic environments. The proposed adaptations were also applied in du Plessis and Engelbrecht (2012) to the DynDE algorithm (Mendes and Mohais, 2005), which is a multipopulation approach. Again, the obtained algorithms were found competitive regarding to other from literature, such as DynDE, jDE (Brest et al., 2009), and mQSO (Blackwell and Branke, 2006).

In Wang et al. (2011), a self-adaptive extension of particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995) is applied to solve the dynamic economic dispatch (DED) problem. This PSO extension includes a self-adaptive technique for adjusting the velocity vector during the run, and was compared against others methods on several instances of the DED problem. The results confirmed the superiority of the self-adaptive variant.

The self-adaptive cluster-based DE with external archive proposed by Halder et al. (2011) is another interesting approach for solving complex DOPs. In this case, an intelligent strategy for redistributing the number of clusters (sub-populations) is employed. However, from the description given by the authors, the proposed strategy seems to be adaptive instead of self-adaptive. Despite of this, they reported remarkable results on the GDBG benchmark.

On the other hand, Chun-Kit and Ho-Fung (2012) studied the performance of three variants of the CMA-ES metaheuristic. Specifically, the authors considered the non-elitist (μ, i)-CMA-ES (Hansen and Ostermeier, 2001), the elitist (1 + 1)-CMA-ES (Igel et al., 2006), and the sep-CMAES (Ros and Hansen, 2008). As baseline for the comparison, they used the adaptive (1 + 1)-ES. The four algorithms were tested on GDBG’s problem instances. The authors also included other scenarios by varying the change severity and the dimension of the search space. From the results, they concluded that elitist variants ((1 + 1)-CMA-ES
and even the simple \((1 + 1)-\text{ES}\) showed a clear advantage over the non-elitist ones. However, in higher dimensions (e.g., > 10), this difference tends to vanish.

In Bahmani-Firouzi et al. (2013), the authors addressed the dynamic economic emission dispatch problem by incorporating wind power plant. To tackle this dynamic multi-objective problem, an improved version of PSO is proposed, which includes a self-adaptive learning strategy for the velocity vector. Such a strategy is used with aims of avoid premature convergence. Compared to other PSO-based algorithms, the self-adaptive one showed an outstanding performance.

Using as test bed a modified MPB with fluctuating numbers of optima, du Plessis and Engelbrecht (2013) analysed the influence of the self-adaptive DE extension of Brest et al. (2006) in a multipopulation approach. Once again, the authors concluded that self-adaptation have important benefits for dynamic optimisation.

An improved self-adaptive GA was proposed by Yang (2013), to tackle a dynamic stochastic shortest path problem. Self-adaptation is expressed here by controlling the crossover and mutation factors. Compared with some very well-known classical algorithms, the proposed method showed a significant performance.

In Arul et al. (2014), a self-adaptive harmony search (HS) was introduced to solve a DED problem with emission and security constraints. Specifically, the HS metaheuristic was extended by including a self-adaptive strategy for adjusting the mutation factor during the run. The reported results, compared to a differential HS algorithm, indicate that self-adaptation has a significant, positive impact in the solution quality, convergence and computational time of the algorithm.

Self-adaptation has been also applied in the context of flocking by swarm robots. This is the case of the self-adaptive communication strategy proposed by Ferrante et al. (2014). Here, the authors give solid evidences that this strategy bring more swarm cohesion than the other non self-adaptive ones.

The performance of niching-based self-adaptive ensemble DE algorithm is analysed by Hui and Suganthan (2014). This algorithm was proposed with aim to solve the problem set given by the CEC’2014 Competition on Dynamic Optimisation. Specifically, the authors extend a self-adaptive variant of DE. The reported results are very promising.

The above works are good examples of advanced methods with self-adaptation at the metaheuristic level, but not with self-adaptation applied to the mechanisms for DOPs. An example of the last case is the strategy proposed by Novoa-Hernández et al. (2013), which added self-adaptation into the diversity during the run mechanism. Specifically, this approach is based on the generation of quantum individuals proposed by Blackwell and Branke (2006). In the original scheme, these random individuals are generated in a ball with a predefined radius \(r_{\text{cloud}}\) during the run. As a consequence, the algorithm tracking ability depends on setting this parameter’s value to a similar magnitude of the shift severity. As the authors pointed out, this is an information that could be unknown in some dynamic scenarios (e.g., real-world problems). In order to solve this issue, they proposed that each DE conventional individual codified in its genotype a \(rc_i\) realisation of parameter \(r_{\text{cloud}}\) being responsible of the generation of one quantum individual. More formally, the conventional individuals are denoted as \(y_i = (x_i, f_i, rc_i)\). In particular, this parameter is mutated as follows:

\[
\begin{align*}
\widetilde{rc}_i & \left\{ \begin{array}{ll}
\text{rand}(0,1) \cdot \lambda \cdot r_{\text{excl}} & \text{if } \text{rand}(0,1) < \tau \\
rc_i & \text{otherwise}
\end{array} \right. 
\end{align*}
\]

where \(\tau, \lambda \in [0, 1]\) are a mutation rate and a scaling factor that are predefined at the beginning of the run, respectively. The \(r_{\text{excl}}\) is the exclusion radius from the original approach (Blackwell and Branke, 2006) and is aimed to limit the exploration area of the subpopulations. As the reader can observe, a new \(\widetilde{rc}_i\) is obtained, with probability \(\tau\), from a variation of the product \(\lambda \cdot r_{\text{excl}}\). Hence, the original \(rc_i\) is kept with probability \(1 - \tau\).

Summarising the revised literature we can note some important facts:

1. There are clearly different opinions regarding the application of self-adaptation in dynamic environments. While most of the works reported successes, there are others in which the considered mechanisms seemed to be not enough to cope with the dynamics of the problem. However, most of these conclusions dated back to the end of ‘90s, in which simple scenarios and algorithms were considered. It would be interesting to find out whether these conclusions are still valid in modern problems and methods.

2. The general approach to apply self-adaptation in dynamic environments is to employ self-adaptive metaheuristics. In other words, the presence of self-adaptation in the algorithm is an inherent ingredient from the employed metaheuristic (e.g., RCGAs, SAES, and JDE). To better understand our point of view, the reader is referred to the scheme of Figure 1. Note that this mentioned case is represented in the scheme by what we have call Class (1) algorithms. As can be seen, self-adaptation appears here as a component of the metaheuristic. Alternatively, there are other works such as (Novoa-Hernández et al., 2013), where the target application is in certain mechanisms for DOPs (e.g., quantum approach). This case is denoted as Class (2) in Figure 1. Finally, there is a third possibility, using self-adaptation in both parts of the algorithm, as it is represented by Class (3), Figure 1. However, as far as we know there is no evidence that this issue has been studied in the past. A study of this type could help to identify the impact of self-adaptation in different scenarios. Having these issues in mind, we have planned the experiments described in the next section.
From Figure 2(a), we can see that self-adaptive extensions of EAs have been more researched than others. Besides, based on our classification [Figure 2(b)], a 81.48% of works employed Class 1 algorithms while none of Class 3. If we focus on the type of research, Figure 2(c) shows that a 70.73% of the papers are empirical ones while just a few devoted to theoretical issues and real applications. Similarly, in the case of type of paper [Figure 2(d)], a 70.73% of them are conference papers or book chapters, being the rest journal articles.

3 Analysis of the role of self-adaptation in dynamic environments

The experiments of this section are devoted to analyse the role of self-adaptation in dynamic environments when it is applied to different parts of the algorithm. As was mentioned before, this remains as an open issue, especially for Class (2) and Class (3) algorithms.

Related to the classical metaheuristic, we have considered three alternatives:
1 Standard DE (Storn and Price, 1997), without self-adaptation.
2 jDE (Brest et al., 2006), DE version with self-adaptation in parameters $C_r$ and $F$.
3 SspDE (Pan et al., 2011), DE version with self-adaptation in parameters $C_r$ and $F$, and the mutation strategy.

Regarding the mechanism for DOPS, we considered the quantum mechanism (Blackwell and Branke, 2006) used to improve the diversity during the run. Then, three alternatives are considered:
1 None, the mechanism is not present.
2 The original mechanism, as was proposed by Blackwell and Branke (2006).

Figure 1 Classes of self-adaptive algorithms in dynamic environments (see online version for colours)

Figure 2 Distribution of the reviewed papers on self-adaptation in dynamic environments according (a) the employed metaheuristic, (b) self-adaptation class, (c) type of research, and (d) type of the paper (see online version for colours)
In general, all methods were implemented following the same template (Algorithm 1), but varying the kind of metaheuristic and the mechanism for DOPs according to the above alternatives. This template is based on the work of Blackwell and Branke (2006) used in the context of PSO. Besides, it is important to highlight that the mechanism used here for change detection is very simple: by re-evaluating the best solution at every iteration.

The combination of the above factors, leads to the algorithms described in Table 1. Note that names of algorithms indicate the presence of the factors. For example, algorithm m+Q+DE is a multipopulation one (m), with the original quantum approach (Q), and the standard DE. Additionally, we have organised the methods into four main groups (first column of the Table 1). First, the baseline, represented by algorithm m+Q+DE is the most basic alternative, that is, without self-adaptation. Secondly, Class (1) composed by algorithms with self-adaptation only at the metaheuristic level. Within this group, we also distinguish a special Class (1)* which is composed by similar algorithms, but with the original quantum approach. The Class (2) contains just the algorithm m+SQ+DE, with self-adaptation only at the level of the mechanism for DOPs. Finally, the self-adaptation in both parts of the algorithm, belong to the Class (3). These are the most sophisticated methods.

### Algorithm 1

General template for all implemented algorithms

```plaintext
1. for each subpopulation \( P_i \) do
   2. Random initialise \( P_i \) as the used metaheuristic (DE, jDE, or SspDE) does;
   3. end
   4. while not stopping condition do
      5. if not change is detected then
         6. for each subpopulation \( P_i \) do
            7. Evolve conventional individuals according the used metaheuristic (DE, jDE, or SspDE);
            8. Generate quantum individuals according the selected approach (none, original, or self-adaptive);
            9. Update best solution of \( P_i \);
         end
      10. else
      11. for each population \( P_i \) do
         12. Reevaluate the \( P_i \) conventional individuals;
      end
      13. end
   14. end
```

We have selected the GDBG (Li and Yang, 2008) as the test bed, specifically the rotation and composition generators employed by the CEC’2009 competition. Both generators comprise 49 instances of problem, which are derived from the combination of seven different landscapes and seven types of changes. Every landscape has ten peak functions, with exception of problem F1(50) which uses 50 peaks. We refer to the landscapes as follows:

1. rotation peak function (F1)
2. rotation peak function with 50 peaks (F1(50))
3. composition of Sphere’s function (F2)
4. composition of Rastrigin’s function (F3)
5. composition of Griewank’s function (F4)
6. composition of Ackley’s function (F5)
7. hybrid composition function (F6).

Specifically, the landscape F6 is obtained by the combination of functions: Sphere, Rastrigin, Grie wank, Ackley and Weierstrass (i.e., two of each one). Regarding the change types, the GDBG proposed the followings:

1. small step (T1)
2. large step (T2)
3. random (T3)
4. chaotic (T4)
5. recurrent (T5)
6. recurrent with noisy (T6)
7. dimensional (T7).

The change type T7 varies the search space dimensions \( D \) in the range [5, 15]. In general, for every instance of problem, the search space is defined as \( \Omega = [-5.0, 5.0]^D \). The environment changes (60 for each execution) occur every \( D \cdot 10,000 \) function evaluations without algorithm knowledge about it. Other details about the parameters settings of this benchmark can be found in Li et al. (2008).

To evaluate the algorithms, we have used as performance measure the error before the change (see Li and Yang, 2008; Li et al., 2008; Novoa-Hernández et al., 2013):

\[
e_{bc} = \frac{1}{NC} \sum_{c=1}^{NC} (f_{opt}^{(c)} - f_{best}^{(c)})
\]

where NC is the number of changes in the problem, while \( f_{opt}^{(c)} \) and \( f_{best}^{(c)} \) are the function values for the problem global optimum and the best solution of the algorithm before the change \( c \), respectively.

In all experiments, we performed 20 runs for every pair algorithm/problem instance with different random seeds.
Table 1  Selected algorithms in our study

<table>
<thead>
<tr>
<th>Cases</th>
<th>Algorithm</th>
<th>Part of the algorithm to be self-adapted</th>
<th>Classical metaheuristic</th>
<th>Mechanism for DOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Classical (1)</td>
<td>m+JDE</td>
<td>JDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Classical (1)*</td>
<td>m+Q+JDE</td>
<td>JDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class (2)</td>
<td>m+SQ+DE</td>
<td>SspDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class (3)</td>
<td>m+SQ+JDE</td>
<td>SspDE</td>
</tr>
<tr>
<td>Baseline</td>
<td>m+Q+DE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Class (1)</td>
<td>m+JDE</td>
<td>JDE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>m+SspDE</td>
<td>SspDE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Class (1)*</td>
<td>m+Q+JDE</td>
<td>JDE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>m+Q+SspDE</td>
<td>SspDE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Class (2)</td>
<td>m+SQ+DE</td>
<td>-</td>
<td>Self-adaptive quantum</td>
<td>-</td>
</tr>
<tr>
<td>Class (3)</td>
<td>m+SQ+JDE</td>
<td>JDE</td>
<td>Self-adaptive quantum</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>m+SQ+SspDE</td>
<td>SspDE</td>
<td>Self-adaptive quantum</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: The methods are derived considering what is the self-adaptation target (classical metaheuristic or mechanism for DOPs).

3.1 Results and statistical analysis

The experiment results are summarised in the Figure 3. In these graphics, the positions (ranking) of the algorithms are showed through different colours, which in turn are based on their corresponding averages of $e_{bc}$.

Every graphic corresponds to a given problem and considering different types of change (as rows). As shown by the chart legend, the darker colours are associated with better ranks (e.g., near to 1), while the worst ranges (e.g., near to 8) are related to lighter shades. We have also included in Figure 3(h) the average ranks of algorithms considering their problem specific ranks. It is important to emphasise that, these ranks are purely descriptive data, that is, they do not provide or belong to any information related with a statistical test.

From these graphics, we can see that algorithms achieve different ranks in the problem instances. However, one can also observe as general tendency, the aggregation of better ranks to right zone of the graphics. The corresponding algorithms in this case are those with self-adaptation in quantum approach (SQ) (e.g., Class (2) and Class (3)). The remaining methods are only partially better in certain problem instances. For example, note the outstanding results of algorithm m+SspDE in Figure 3(d), its associated column of ranks is the darkest.

Other interesting result is that the baseline algorithm m+Q+DE outperforms in some cases the self-adaptive variants from Class (1) [Figure 3(e)]. The later and its special Class (1)* are only better for a few scenarios [Figure 3(g)]. If we constrain our comparison to Class (1) and Class (1)*, it is possible to observe that it is difficult to identify which method is the best. For example, while m+jDE and m+SspDE are worse than m+Q+jDE and m+Q+SspDE in scenarios from Figures 3(a) to 3(c), there are scenarios where this superiority is not observed [Figures 3(d) and 3(g)], or simply hard to detect.
Self-adaptation in dynamic environments – a survey and open issues

Figure 4  Average ranking of the algorithms according to the Friedman test, (a) unimodal peak functions (b) multimodal peak functions (c) all functions (see online version for colours)

Note: The arcs tying two algorithms indicate that there are not differences between them (according the adjusted $p$-values from the Holm’s post-hoc test.}

In order to formally check if the previous differences are meaningful from a statistical point of view, we performed non-parametric tests using as dataset the averages values of the error before the change reported by algorithms. We divided the analysis in three major groups, according the nature of the peak function used by the instance of problem: unimodal, multimodal and all functions.

First, a Friedman test was applied to detect differences at group level. The corresponding $p$-values were lower than 0.05 in the three cases. So, we proceeded with a post-hoc Holm’s test in order to detect which pairs of the algorithms are different. These results are graphically represented in Figure 4. Note that algorithms are nodes, which are tied by arcs if no significant difference exists in the corresponding pair. Figure 4 also shows the positions obtained by the algorithms according the average ranks of the Friedman test. The lower rank, the better is the algorithm.

If we consider the first group of results, we can see that for scenarios with unimodal peak functions, the best performing algorithms are those from Class (3) and Class (2). However, algorithms $m+SQ+DE$ and $m+SQ+jDE$ are not significantly different with respect to those from Class (1)*. In particular, the worst algorithms are those with self-adaptation only at metaheuristic level.

Regarding multimodal peak scenarios, we can see slightly different results than the previous group. For example, Class (2) and Class (3) algorithms are the best, but interestingly algorithm $m+SspDE$ has a good performance. As discussed above, this is mainly because the great performance of this algorithm in problem instances with the Rastrigin peak function [see Figure 3(d)].

3.2 Analysis of the algorithm adaptability

We will study the contribution of the considered self-adaptive mechanisms to the algorithms from a different point of view. Specifically, we are going to compare self-adaptation at metaheuristic level versus self-adaptation at the mechanism for DOPs. For this purpose, we select the following algorithms:

- $m+jDE$ (Class 1)
- $m+SspDE$ (Class 1)
- $m+SQ+DE$ (Class 2)

because the other ones are hybrids and therefore would be more difficult to discern whether the results depend on the presence (or absence) of other elements.

We need to define first what goal has these schemes in the context of dynamic environments. One might argue that the role of these schemes is to maintain a proper balance between exploitation and exploration. For population-based algorithms, the above requirement is somehow equivalent to change the diversity of the population in such a way to put it in correspondence with the state of the search. For example, if the optimum is close, the level of diversity should be low.

Therefore, it can be deduced that self-adaptation in dynamic environments should change the population diversity as a function of the distance to the optimum of the problem. Based on this reasoning, we proposed the following expression to measure the level of adaptability of the algorithms:

$$adaptability^{(t)} = relDistOpt^{(t)} - relDivPob^{(t)}$$

where $relDistOpt$ and $relDivPob$ are normalised magnitudes related with distance to the optimum of the problem, and the population diversity, respectively. The normalisation was made taking into account the best and the worst values from the considered algorithms. These magnitudes were defined in Weicker (2002), and Deb and Beyer (2001) as indicated below. The distance to the optimum is:
\[ \delta_{opt} = \sqrt{\sum_{d=1}^{D} (x_{opt}^d - x_{best}^d)^2} \]  

where \( D \) is the dimension of the search space, while \( x_{opt}^d \) and \( x_{best}^d \) are the \( d \)th component of the optimum solution and the best solution of the algorithm, respectively.

Regarding the population diversity, it is given by the average variance of the population with respect to the population centroid:

\[ d_{\text{pop}} = \frac{1}{\mu} \sum_{i=1}^{\mu} \left( \frac{1}{D} \sum_{d=1}^{D} (x_{\text{cent}}^d - x_i^d)^2 \right) \]

where \( \mu \) is the population size, while the \( x_{\text{cent}}^d \) and \( x_i^d \) are the \( d \)th component from the population centroid and the \( i \)th individual, respectively. The centroid is defined by the vector \( x_{\text{cent}} \) with components computed as follows:

\[ x_{\text{cent}}^d = \frac{1}{\mu} \sum_{i=1}^{\mu} (x_i^d) \]

Both \( relDistOpt \) and \( relDivPop \) are defined in the range \([0, 1]\), so \( \text{adaptability} \in [-1, 1] \). If \( \text{adaptability} \) approaches 1, that means that the algorithm is unable to generate enough diversity to follow the optimum. Conversely, a value close to \(-1\), indicates that the algorithm generates more diversity than necessary. Hence, the ideal value would be 0, which means that diversity is changed according to the distance to the optimum.

**Figure 5** Adaptability and accumulated adaptability during the run of algorithms m+SQ+DE, m+jDE and m+SspDE, (a) adaptability (function F1) (b) accumulated adaptability (function F1) (c) adaptability (function F3) (d) accumulated adaptability (function F3) (see online version for colours)

Notes: Peak function is F1 on plots (a) and (b) and F3, on (c) and (d). Change type is large step in both cases.
Figure 5 plots the adaptability of algorithms on problems with peak functions rotation peaks and composition Rastrigin, and a large step change type. In particular, plots 5(a) and 5(c) show the adaptability as a function of algorithm iterations, while the other plots 5(b) and 5(d) display the accumulated average of these values. These last plots have been employed to visualise the general trend of this measure. Please note that the changes in the environments occur after 1,000 iterations (see x-axis).

As the reader can see, the behaviour of the algorithms is consistent with the previous statistical analysis. For example, the algorithm m+SQ+DE shows a better adaptability than the rest in the scenario with peak function F1 (plot 5-b). Nevertheless, observing plot 5-a), one may conclude that the three algorithms show self-adaptive behaviours in general, because they generate diversity after the changes (every 1,000 iterations of the algorithms).

It is clear that m+SQ+DE (Class 2) achieves a faster and more effective adaptation. Note how fast the accumulated adaptability of this algorithm approaches to 0 when time takes greater values. In the case of scenario with function F3, such self-adaptive behaviour is not so easy to observe as in the previous case [graphic 5(c)]. Undoubtedly, the complexity of the F3 function (composition Rastrigin) negatively affects the adaptability of all algorithms that is why the similar trends among them [graphic 5(d)]. However, at least for the 15 changes considered in the plot 5(d), a very slow decreasing tendency in all cases can be observed.

4 Conclusions and future works

In this work, we survey the current research on self-adaptation in DOPs. From this review, we distinguish three classes of algorithms according the level of application of self-adaptation. While the first one (i.e., self-adaptive metaheuristics designed for stationary environments) has been well studied in the past, the other two classes have not (i.e., self-adaptation at the mechanism for DOPs, and the combination of both). In order to explore these open issues, we empirically analysed the performance of eight multipopulation algorithms based on the DE metaheuristic. Specifically, the algorithms were obtained from the use or not of self-adaptation models at the metaheuristic level and/or at the mechanism used to deal with DOPs.

The experiment conducted on 49 instances of problems with different peak function and changes type can be summarised as follows:

1. using self-adaptation only at metaheuristic level is not enough to deal with the dynamism of the problem
2. self-adaptation applied to a mechanism for DOPs (e.g., diversity during the run) leads to significant better algorithms
3. algorithms with self-adaptation at both levels are significant better than those with self-adaptation at the metaheuristic level, but not than those using it only at the level of mechanism for DOPs.

In order to better understand the reasons of such conclusions, we proposed a measure to study the adaptability of the algorithms. The results showed that algorithms with self-adaptation at the metaheuristic level show less adaptability than the algorithm with only self-adaptation applied at the mechanism for DOPs.

Despite the usefulness of the results achieved in this research, we consider that this topic needs further attention because we envisage that self-adaptation is the way to go in order to obtain more ‘intelligent’ and effective algorithms for dynamic environments.

As a future works, we plan to analyse other self-adaptive models from stationary optimisation in the context of dynamic environments. The idea here is to verify whether the conclusions of the present research still hold for other metaheuristics and mechanisms for DOPs.

Acknowledgements

P. Novoa-Hernández has the support of a post-doctoral scholarship from the Eureka SD project (Erasmus Mundus Action 2) coordinated by the University of Oldenburg, Germany.

C. Cruz Corona and D.A. Pelta acknowledge support from projects TIN2011-27696-C02-01, Spanish Ministry of Economy and Competitiveness and P11-TIC-8001 from the Andalusian Government (including FEDER funds from the European Union), and GENIL-PYR-2014-9 project from University of Granada, Spain.

References


Notes

1 http://www.scopus.com/.