On the elusivity of dynamic optimisation problems.

ALZA, J., BARTLETT, M., CEBERIO, J. and MCCALL, J.

2023

On the Elusivity of Dynamic Optimisation Problems

Joan Alza[a,b,*], Mark Bartlett[b], Josu Ceberio[c], John McCall[a,b]

[a]*National Subsea Centre, 3 International Avenue, Aberdeen, AB21 0BH, Scotland, United Kingdom*
[b]*Robert Gordon University, Garthdee Road, Aberdeen, AB10 7GJ, Scotland, United Kingdom*
[c]*University of the Basque Country, Manuel Lardizabal 1 Pasealekua, San Sebastian, 20018, The Basque Country, Spain*

Abstract

The field of dynamic optimisation continuously designs and compares algorithms with adaptation abilities that deal with changing problems during their search process. However, restarting the search algorithm after a detected change is sometimes a better option than adaptation, although it is generally ignored in empirical studies. In this paper, we suggest the *elusivity* formulation to (i) quantify the preference for restart over adaptation for algorithms running on dynamic problems, and (ii) evaluate the advantage and behaviour of adaptation. Informally, we state that a dynamic problem is *elusive* to an algorithm if restart is more effective than adapting to changes. After reviewing existing formalisms for dynamic optimisation, the elusivity concept is mathematically defined and applied to two published empirical studies to evaluate its utility. Conducted experiments show that replicated works include elusive problems, where restart is better than (or equal to) adaptation, and demonstrate that some empirical research effort is being devoted to evaluating adaptive algorithms in circumstances where there is no advantage. Hence, we recommend how and when elusivity analysis can be gainfully included in empirical studies in the field of dynamic optimisation.

Keywords: Dynamic Optimization Problem, Elusivity, Adaptative Advantage, Online Solving, Restart

*Corresponding author.
Email addresses: `j.alza-santos@rgu.ac.uk` (Joan Alza), `m.bartlett3@rgu.ac.uk` (Mark Bartlett), `josu.ceberio@ehu.eus` (Josu Ceberio), `j.mccall@rgu.ac.uk` (John McCall)

1. Introduction

The optimisation community identifies dynamic problems as optimisation problems that change at least once during the execution of an algorithm. They are commonly instantiated as sequences of static problem instances, where the problem varies during algorithm execution. Under this set-up, the literature presents various benchmark generators for creating dynamic problems from static optimisation problems [1, 2, 3, 4]. Starting from a static problem instance, these methods produce a sequence of static problem instances by incrementally making changes to the latest problem instance in the sequence.

Although dynamic problems may possess many features, such as the time-linkage of changes [5], the dynamism is often reduced to the calibration of two prominent features: the frequency and magnitude of changes [6]. In general, it is assumed that the more frequent and severe the change, the more challenging the problem becomes for an algorithm dealing with changes. For example, a dynamic routing problem where multiple orders arrive simultaneously is likely to be more difficult to an algorithm than a problem where a single order is added from time to time. However, it may be the case that algorithms perform efficiently on problems that change with high frequency or magnitude [6].

In academia, a *dynamic optimisation problem* (DOP) is defined as a special class of dynamic problems that are solved *online* by an optimisation algorithm as time progresses [5]. As mentioned in [7], solving the problem online means that the dynamic problem must be solved as time goes by to approach such problems. The research community has developed a variety of powerful online algorithms able to deal with changes in the problem during algorithm execution [8]. These algorithms typically contain adaptation mechanisms aimed at improving the overall algorithm performance [9].

The literature presents a number of works that study the impact of problem changes on algorithm performance. Branke [10] proposes that optimisation problems should be considered *dynamic* only if the algorithm adapts accurately to the changes over time. Conversely, problems that vary without reference to previous stages should be regarded as sequences of independent problems. The authors in [11] present some measures to analyse and characterise the nature of changes based on the shifting distance of the best solutions after a change in the continuous domain. Rohlfshagen et al. [6] theoretically analyse the runtime of a (1+1) evolutionary algorithm on two simple frameworks according to the magnitude and frequency of changes, and illustrate two counter-intuitive assumptions about the dynamic domain where restart is preferred over adaptation for slightly changing problems. Further theoretical works also analyse the runtime analysis of different algorithms for different well-studied dynamic problems [12, 13, 14, 15]. Yu et al. [16] demonstrate that relocating the moving optima may be difficult in severely and quickly changing continuous optimisation problems. Tínos and Yang in [17] give realistic examples where solving the instances independently (restarting the algorithm after detecting a change) is effective, and they suggest that in those cases dynamic problems should be considered as series of unrelated static instances. Similarly, the authors in [18] empirically show that restarting the algorithm is often the best option on a drug mixture problem with severe drug replacement rate. The authors in [19] review the field of dynamic continuous optimisation, and point out that a prominent future direction is the analysis of algorithms performance and capability of online approaches on dynamic problems with different dynamics.

As far as we are aware, there is no systematic quantitative analysis that measures and relates problem variation, adaptability of algorithms and algorithm performance. Hence, the motivation of this work is to be able to accurately quantify the extent to which an online algorithm successfully adapts to a dynamic problem, given a performance metric. In other words, this work aims to quantify to what extent an online algorithm successfully adapts to changes compared to a restart under a performance metric. Assuming that changes are known or easily detectable by the algorithm, restarting the search is probably the simplest and most straightforward approach to tackle problem changes, and it could also be valuable when reusing old information does not provide any advantage [20, 21]. However, most empirical studies in dynamic optimisation generally operate under the implicit assumption that adapting to *small* or to *medium* changes is generally more efficient and stable than restarting the search from scratch [22].

In this paper, we define the *elusivity* that quantifies and analyses the degree to which adapting to a problem change is beneficial, for an online algorithm, rather than restarting the search from scratch (after detecting a change) given a specific dynamic problem and a performance metric. Moreover, we suggest using the restart as the baseline against which all adaptation mechanisms should be compared to evaluate its adaptative advantage.

The contribution of this work is to precisely formulate the elusivity in mathematical terms, and apply the metric to (i) measure the extent to which adaptation improves or degrades algorithm performance in comparison to a restart after a change, and (ii) use the elusivity to study the advantage and behaviour of adaptation mechanisms.

In order to illustrate and evaluate the utility of the metric, we apply elusivity analysis to two already published empirical studies to evaluate the overall elusivity of the algorithms considered, i.e. in what cases a simple restart is more effective than the implemented online algorithms [2, 22, 23]. Conducted experiments use well-known dynamic benchmark generators, metaheuristic algorithms and performance metrics. Obtained results demonstrate the existence of some elusive problems in the reproduced experiment, where adaptation mechanisms confer little or no advantage over restart for specific problem, algorithm and performance metric combinations. That is, not only severe changes prove elusive, but frequently and slightly changing problems may also be elusive to some algorithms under a performance metric. In these cases, adaptation mechanisms confer little to no advantage, or are even disadvantageous. This shows that the counter-intuitive behaviour exhibited in the theoretical example of [6] is also observable in mainstream state-of-the-art algorithms operating on widely-used benchmarks. While in no way invalidating published works, our results allow us to distinguish the cases in which online algorithms are performing successfully on problems from cases where adaptation mechanisms are not operating effectively. That is, we distinguish elusive and non-elusive problems based on the adaptative advantage of algorithms. In addition, by examining the elusivity distribution of problems with different magnitudes and frequencies of change, we obtain information about the performance and behaviour of the adaptation mechanisms to the configurations used.

The remainder of the paper is structured as follows. Section 2 presents a brief review of properties of dynamic benchmark generators and dynamic problems in the literature. Section 3 introduces the elusivity formulation and its mathematical formulation. Section 4 describes in detail the empirical case studies considered in this work. Section 5 is dedicated to the analysis and visualisation of results. Section 6 summarises the implications of the observed results and concludes the paper by presenting valuable research streams and future application guidelines.

2. Dynamic Optimisation Problems and their Properties

Dynamic problems have been widely defined as *sequences of static problem instances* [24, 3, 25, 26], although others defined them as *optimisation problems parametrised by time* [27, 28, 7, 29]. Nguyen in [5] provides a theoretical basis, and defines dynamic optimisation problems (DOPs) as *dynamic problems that are solved online by an algorithm as time goes by.* Informally, assuming that time is a discrete parameter, solving the problem *online* means that at time t^{now}, the function cannot be evaluated at time $t > t^{now}$ [7].

Researchers typically build dynamic test problems using benchmark generators that insert changes into the objective function, the problem instance, the decision variables or the constraints of existing static problems [8]. The literature presents some classification schemes based on the characteristics of changes. These classifications intend to taxonomise the difficulty of algorithms to solve the problems. The authors in [29, 26] present similar classification schemes that can be summarised as follows:

- Eberhart and Shi [31] differentiate three categories based on the *change direction* in the continuous space, i.e. whether a change modifies the location of an optima in the fitness landscape, its objective value, or both.

- Presented in the continuous domain, the *trajectory* of the optima describes the path of the moving optima in the changing fitness landscape. Angeline [32] distinguishes three types of changes: linear, cyclic and random. The linear dynamism maintains a constant displacement of the optima considering a change magnitude. Cyclic problems shift the optima through a circular path considering a magnitude as the radius of the circular path. The random dynamism add certain variation to the objective function every time it is changed.

- Introduced by De Jong [33] and extended in [30], dynamic problems may be categorised based on their most prominent parameters: the frequency and magnitude of changes. The four categories proposed by the authors are: quasi-static problems (slightly and occasionally changing problems), progressive problems (frequently and slightly changing problems), abrupt problems (occasionally and significantly changing problems), and chaotic problems (drastically and frequently changing problems). Figure 1 displays graphically the above mentioned taxonomisation according to Duhain and Engelbrecht in [30].

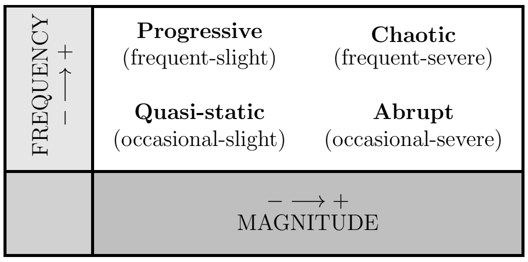


Figure 1: Classes of problems presented in [30].

In empirical studies, it is common to compare online algorithms across benchmark sets designed to provide varied challenges across the enumerated categories. However, these studies implicitly assume that restart is an inferior option over adaptation for small or medium changes [1], although it is not always given, i.e. it varies according to the algorithm used. In the next section, we present a formulation to quantitatively measure the preference for restart over adaptation for algorithms running on a dynamic problem: the elusivity of a problem to an algorithm under a performance metric.

3. Elusivity of dynamic problems

In this section, we define *elusivity* as a quantitative measure of the performance difference between the algorithm that adapts to changes against the performance that would have been achieved by the same algorithm restarting the optimisation when the problem is changed. We assume that algorithms are able to detect when a change happens in the problem. Therefore, in the event that restarting the algorithm is better, or no worse, than dealing with problem changes, we can observe that the algorithm cannot adequately adapt to changes: the problem proves *elusive* to the algorithm.

To make this precise, we introduce some mathematical terminology to describe the performance of the aforementioned (online and restarting) algorithms and define a formulation of the elusivity of a problem to an algorithm. In the definitions that follow, we aim to describe dynamic problems and algorithms in sufficient generality to apply to most situations of interest, and avoid unnecessary assumptions about problems or algorithms. Therefore, we draw on the language of stochastic processes to abstract any detail of the internal state of algorithms into random variables representing successive states of the search. In applying these ideas to specific algorithms, internal state parameters are known and the extent to which they condition search state transitions can be estimated experimentally.

Simply, a *static optimisation problem* $P_s = (X, f)$ can be defined as a tuple consisting of a search space X and an objective function $f : X \to \mathbb{R}$. This notion captures any representation of solutions, and it can also be extended to multiple objectives without difficulty.

A search algorithm A is *run on a static problem* $P_s = (X, f)$ with budget B whenever A evaluates solutions in X until reaching a maximum of B evaluations (or iterations) using f. We can represent a *run* of the algorithm A on problem P_s with budget B as a sequence of random variables $S_1(A, P_s), \ldots, S_i(A, P_s), \ldots, S_B(A, P_s)$. The value that each random variable takes in a single run is a sample of one or more solutions in X, called *populations.* Note however that this notation is not limited to only population-based algorithms, since other structures can be used in essentially the same way. We can write $Pr(S_i = s)$ for the probability that s is the i^{th} *population* generated by the algorithm A, i.e. during a run on the static problem P_s, A generates a sequence of populations $s_1, \ldots, s_B$. In the remainder, we refer to these random variables as *populations*, and the sequence of populations as a *trajectory*[1] of algorithm A on problem P_s.

Algorithms start from a given distribution of initial populations, usually uniform $U_{(A,P_s)}$[2]. Following our notation above, the initial random variable on the run of algorithm A on a static problem P_s is drawn from

$$S_1(A, P_s) = U_{(A,P_s)}. \tag{1}$$

Since the trajectory is generated by sampling a sequence of random variables, it is itself a random variable. We denote by $Tr(A, P_s)$ the random variable of trajectories of algorithm A on problem P_s, and will refer to $tr(A, P_s)$ to denote a specific value sampled from $Tr(A, P_s)$. That said, we can also regard a given trajectory $tr(A, P_s)$ as “a run of algorithm A on problem P_s”.

[1]For non-population-based algorithms, such as local search algorithms using candidate solution(s) as structure, their structure can easily be added to the *trajectory* without altering our formulation.

[2]This statement however could equally apply to algorithms starting with a predefined initialisation strategy by replacing $U_{(A,P_s)}$ with the approach used to generate the initial random variable.

A performance metric $\phi(A, P_s)$ can be defined as a function which assigns a real number to a run $tr(A, P_s)$ of the algorithm A on problem P_s. In other words, ϕ is a function from run trajectories to real numbers. Since $Tr(A, P_s)$ is stochastic, the performance of algorithms over R runs can be used to estimate the expected performance $\mathbb{E}[\phi(A, P_s)]$.

In order to extend previous definitions to the dynamic domain, we need to formulate dynamic problems and online algorithms precisely. A dynamic problem P can be defined as a sequence of static problem instances $P_1, ..., P_m$, where $P_j = (X_j, f_j)$ for each $j = 1, \ldots, m$, and P_j is defined on the space X_j and the objective function $f_j : X_j \to \mathbb{R}$. To ensure the problem is changed at least once during the optimisation run, we set $m \geq 2$ to mean $m-1$ objective function changes.

Empirical benchmark studies in dynamic optimisation assume that online adaptation is usually beneficial to track efficiently the moving optima and reduce the computational cost. Informally, a search algorithm A is run online on the dynamic problem P with budget $B = \sum_{j=1}^{m} B_j$ means that A outputs a related trajectory $tr(A, P)$ while successively processing a sequence of problem instances P_j. Using the above notations, we can say that the algorithm A is conditioned on the trajectory $tr(A, P_{j-1})$ through changes in the objective function f_j, i.e. the i^{th} population of A in P_j is sampled from $S_i(A, P_j)$, which is conditional on the so far trajectory of A.

These notations allow us to understand how an online algorithm runs on a dynamic problem (previously introduced as DOPs), and also to make precise what we mean by restarting the algorithm for dynamic problems. Assuming that changes are detectable, we define A^r as the algorithm A running with restart on problem P with budget $B = \sum_{j=1}^{m} B_j$ to mean that A runs *independently* with budget B_j on each of the instances P_j, i.e. the overall trajectory $tr(A^r, P)$ is a concatenation of the m independent trajectories $tr(A, P_1), \ldots, tr(A, P_m)$, where the initial population of each instance is initialised as $S_1(A, P_j) = U_{(A,P)}$. In other words, A^r means A running from initialisation on P_1 to generate $tr(A, P_1)$, A running from initialisation on P_2 to generate $tr(A, P_2)$, and so on.

It is worth noting that, in general, $S_i(A, P_j)$ and $S_i(A^r, P_j)$ will follow different probability distributions, determined by the different trajectories of A and A^r, respectively. In this way, we can compute and compare trajectories and performance metrics for A and A^r over general dynamic problems. Without loss of generality, we make the choice that the algorithmic goal is to minimise a performance metric. This prompts the following definitions.

Definition 1. *Let P be a dynamic problem, A be an algorithm running online on P and ϕ be a performance metric. We define the* elusivity *of P to A under ϕ as:*

$$\mathcal{E}(P, A, \phi) = \mathbb{E}[\phi(A, P) - \phi(A^r, P)].$$

Definition 2. *We say that the problem P is* elusive *to the algorithm A under the performance metric ϕ iff $\mathcal{E}(P, A, \phi) \geq 0$.*

Informally speaking, if P is elusive to A under ϕ, then there is no performance advantage to be gained from A as the problem changes. In other words, equally good or even better performance can be expected from simply restarting the algorithm when detecting a problem change. This establishes restart as the baseline against which all online algorithms should be compared with, i.e. online algorithms must, at least, beat their restarting version to mean that we work with *non-elusive* problems. Trivially from our definitions, all dynamic problems will be elusive to algorithms that adapt by re-initialising the entire population (not necessarily uniformly at random), because in this case we have $Tr(A^r) = Tr(A)$. Moreover, the elusivity provides us with a measure of how well or badly a particular algorithm adapts to a particular dynamic problem, allowing us to quantitatively compare algorithms over dynamic benchmark sets. Strongly negative elusivity will indicate that an algorithm is well-suited to a particular benchmark set. Finally, the concept enables us to quantitatively relate the online performance to problems and algorithm features.

4. Case Studies for Elusivity Analysis

In this work, we evaluate the elusivity concept by using and extending the analysis on two already published experimental works, i.e. a number of well-studied non-noisy dynamic problems, algorithms and performance metrics are re-implemented from two existing frameworks, and the resulting trajectories are subjected to elusivity analysis. In that context, there are a number of critical elements that we address in this section: (i) the dynamic generators used to insert dynamism into static problems, (ii) a brief review of the algorithms, the adaptation mechanisms incorporated to enhance their adaptability and their restarting version, and (iii) the functions used to measure the performance of the algorithms.

4.1. Dynamic Benchmark Generators

Probably, the most studied method to generate artificial dynamic testing problems from any existing static combinatorial problems is the landscape rotation [2, 3, 4, 34]. Its popularity results from its simplicity and capability to maintain certain properties of the static problem, such as the number of optima or the objective function values. However, it is usually criticised because of its unrealistic nature [25, 35].

Characterised by the period τ and magnitude ρ of changes, the generator constructs artificially a concatenated homogeneously changing problem by disrupting the mapping between solutions and objective values over time. Formally, given a static problem $P_s = (X, f)$, the dynamic problem P is represented by

$$f_j(\boldsymbol{x}) = f(\boldsymbol{M}_j \cdot \boldsymbol{x}),$$

where f_j is the changing objective function in $P_j = (X, f_j)$ and $\boldsymbol{M}_j$ is a binary mask. The mask is incrementally generated by $\boldsymbol{M}_{j+1} = \boldsymbol{M}_j \cdot \boldsymbol{T}$, where $\boldsymbol{T}$ is a template generated uniformly at random containing $\lceil n \times \rho \rceil$ variations. The problem is periodically changed based on the budget $B_j = \lceil \frac{i}{\tau} \rceil$, subject to the iteration $i = 1 \ldots B$ and the change period τ. It is worth noting that there are some components of the generator that vary with representations as shown in Table 1. For more information, we direct the interested reader to [2, 3, 34].

Another common way to insert dynamism into a static problem is modifying the initial problem instance parameters. In [23], the authors associate these dynamics with the traffic factor in the dynamic travelling salesperson problem (DTSP), and present a benchmark generator based on

$$f_j(\sigma) = d_j(\sigma_1, \sigma_2) + \sum_{i=2}^{n} d_j(\sigma_i, \sigma_{i+1}),$$

where $d_j(x, y)$ is the distance between cities x and y for instance j. So following the notation above, the search space X remains constant over the m instances, whereas the varying objective function f_j is incrementally modified by

$$d_j(x, y) = \begin{cases} d(x, y) + r, & \text{if } (x, y) \in W_j, \\ d_{j-1}(x, y), & \textit{otherwise}, \end{cases}$$

where $d(x, y)$ is the original distance between cities x and y, $d_j(x, y)$ is the distance between cities x and y on the problem instance j, $r \sim N(0, d(x, y))$ is

Table 1: Components that vary with the class of problem.

Component	**Binary problems**	**Permutation problems**
Mapping function $(\cdot)$	XOR operation $(\oplus)$.	Composition between permutations $(\circ)$.
Change representation $\lceil n \times \rho \rceil$	Number of ones to be flipped (Hamming distance).	Permutation distance metric dependant (relabels for Hamming distance or swaps for Cayley distance, for example).
Initial mask $\boldsymbol{M}_j$	A zero vector, $\boldsymbol{M}_1 = \{0\}^n$.	Identity permutation, $\boldsymbol{M}_1 = e = 12\cdots n$.

a normally distributed random number and W_j is a set of randomly selected arcs for the change period j. In the case of asymmetric DTSPs, the magnitude of changes ρ implies the modification of $\lceil n(n-1)\rho \rceil$ arcs in W_j, where $n(n-1)$ is the total number of connections between cities. For symmetric instances, this number is halved, $\lceil \frac{n(n-1)}{2}\rho \rceil$, since $d_j(x,y) = d_j(y,x)$.

Besides, the authors in [23] also suggest using the city replacement to generate dynamism into static TSPs [36], where a number of cities are replaced with new ones. Formally, given a graph $G = (V, E)$ with v nodes and e edges, this benchmark generator replaces, every τ iterations, $\lceil \rho \times \frac{n}{2} \rceil$ random nodes (cities) in N_{in} in the instance with the same number of random nodes in N_{out}, where V is a set of nodes (cities), E is a set of arcs fully connecting the nodes, τ and ρ are the frequency and magnitude of changes, respectively, n is the size of the problem and $N_{in} \cap N_{out} = N$ are two sets containing $\frac{n}{2}$ nodes each.

4.2. Algorithms and adaptation mechanisms

The algorithms used in this experimentation have been selected from previous works [37, 38, 39, 40], where the benchmark generators above are used. For the first case study, three well-studied algorithms are accompanied by two immigrant-based adaptation mechanisms: the elitism and random immigrants-based approaches [37, 41]. Hence, an elitism and a random immigrants-based genetic algorithm (EIGA and RIGA) [37], population-based incremental learning (EIPBIL and RIPBIL) [2, 38] and ant colony optimisation (EIACO and RIACO) [42] are considered for this case study.

In short, given a replacement rate, the immigrants-based adaptation mechanisms for EIGA, RIGA, EIPBIL and RIPBIL substitute the less promising (worst) solutions of the population with a set of immigrants (solutions) every iteration, no matter whether the problem changes or not. In the case of EIACO and RIACO, the immigrants replace the worst solutions in a memory (set of solutions), rather than replacing the worst solutions in the population [39]. A replacement rate of 1.0 means replacing the entire population with new immigrants. The elitism immigrants-based adaptation mechanism have been proved to be useful for slightly and occasionally changing problems [37, 38, 39], whereas the random immigrants approach is suitable for fast and considerably changing environments [43, 40, 37, 39], *quasi-static* and *chaotic* changes in Figure 1, respectively.

For the second case study, we consider four ant colony optimisation (ACO) variants presented in [23]: two pheromone evaporation-based algorithms (MMAS and MC-MMAS), and two memory-based algorithms (PACO and EIACO). These two types of algorithms, evaporation- and memory-based, differ in the way they update the pheromone-trail, and consequently, in the way they adapt to problem changes. Pheromone evaporation-based algorithms balance the stability of the algorithm by reducing the evaporation trail of previously found good solutions, whereas the memory-based algorithms use approaches to store previously found good solutions, so they decrease faster outdated pheromone trails. We refer the reader to [23] for further details about the algorithms.

The restarting version of the algorithms[3] dismisses the gathered information and re-initialises the algorithm parameters every time the problem is changed. Restarting the optimisation when a change occurs is probably the simplest and most naive way to tackle a problem-change, and it can be useful in some cases [22, 17]. Restart however is usually computationally expensive, in addition to the fact that changes must be detected. In this work, we use *detectors* [9] that reevaluate the best solution of the population each iteration to check if their objective value has been altered, i.e. a change occurs when the objective value of the detector at iteration i and $i+1$ differs.

[3]The superscript "A^r" refers to the restarting version of the algorithm A.

4.3. Performance Metrics

Algorithms are usually evaluated using performance metrics, where different factors (specified by the practitioner) are measured depending on the goals of the problem. For example, the financial field is more focused on obtaining good solutions quickly, rather than finding the best solution possible for each problem instance [9]. As a result, performance metrics can be classified into two classes: optimality-based and behaviour-based performance metrics. The former evaluates the ability of algorithms to find high-quality solutions, whereas the latter studies the internal nature of algorithms, such as their recovery speed or stability. In this work, we considered three optimality-based measures used in the replicated works, and we also consider a behaviour-based measure.

First, the best-of-generation [2] averages the value of the best objective value at each iteration over several runs. Formally, it may be described as:

$$\mathbb{E}[F_{BOG}(A,P)] = \frac{1}{R}\sum_{k=1}^{R}(F_{BOG}(A,P)) = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{1}{B}\sum_{i=1}^{B}x^*_{i,k}(A,P)\right), \quad (2)$$

where R is the total number of independent runs and $x^*_{i,k}$ is the best objective value of the population $s_i(A,P_j)$ at run k. Therefore, the elusivity of a problem to an algorithm under best-of-generation is measured as:

$$\mathcal{E}(P,A,F_{BOG}) = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{1}{B}\sum_{i=1}^{B}\left[x^*_{i,k}(A,P) - x^*_{i,k}(A^r,P)\right]\right). \quad (3)$$

Second, the accuracy [44] averages the difference between the optima and the value of the best individual at the end of each change period. [45] adapted it to average the accuracy over the runs. It may be represented as:

$$\mathbb{E}[H_{\Delta m}(A,P)] = \frac{1}{R}\sum_{k=1}^{R}(H_{\Delta m}(A,P)) = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{1}{m}\sum_{j=1}^{m}h_j(A,P)\right), \quad (4)$$

where h_j is the best-error when the problem instance P_j reaches the budget B_j on run k, and m is the total number of instances. The error is assumed to be the difference between the best-known value of the instance and the value of the best individual in the population. By replacing the best-error h_j with the best found objective value just before changing P_j to P_{j+1}, x^*_j, we obtain the best-before-change performance metric, $P_{\Delta m}$, that is used as the third measure in our experimentation (as done by [23]). The elusivity of a problem to an algorithm under the accuracy measure is calculated as:

$$\mathcal{E}(P, A, H_{\Delta m}) = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{1}{m}\sum_{j=1}^{m}\left[h_j(A, P) - h_j(A^r, P)\right]\right). \tag{5}$$

It is worth noting that, since the accuracy measures the difference between the best found solution's objective value and the optimal objective value, the elusivity analysis of the accuracy or the best-before-change metrics is the same. In other words, the elusivity formulation of $H_{\Delta m}$ and $P_{\Delta m}$ can be summarised as the difference of the best found objective values of the online and restart algorithms, A and A^r, respectively.

Fourth, the robustness [46] measures the stability, persistence and degradation of an algorithm by comparing the best already found solution with the best solution found on the last change (instance) in the following way:

$$\mathbb{E}[\mathcal{R}(A, P)] = \frac{1}{R}\sum_{k=1}^{R}\left(\mathcal{R}(A, P)\right) = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{1}{m}\sum_{j=1}^{m}\min\Big(1, \frac{\Delta x^*_{j-1}(A, P)}{x^*_j(A, P)}\Big)\right), \tag{6}$$

where Δx^*_{j-1} is the best objective value found so far and x^*_j is the best objective value found for instance j. $\mathcal{R} \in [0, 1]$ is a maximisation performance metric, i.e. the closer to 1, the better average robustness of the algorithm A. The `min` operation evaluates if the algorithm is able to reach the same or even a better objective value than previous changes. Note that this equation is prepared for minimisation problems, although it can be easily transferred to maximisation problems by replacing the `min` with the `max` operation. That said, the elusivity of a minimisation problem to an algorithm measured by its robustness is therefore obtained by:

$$\mathcal{E}(P,A,\mathcal{R}) = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{1}{m}\sum_{j=1}^{m}\left[\min\Big(1, \frac{\Delta x^*_{j-1}(A, P)}{x^*_j(A, P)}\Big) - \min\Big(1, \frac{\Delta x^*_{j-1}(A^r, P)}{x^*_j(A^r, P)}\Big)\right]\right). \tag{7}$$

Table 2: Parameter values of the benchmark generators and algorithms extracted from reference works [2, 22, 23, 37, 42].

Parameter	**Case Study I** **Value**	**Case Study II** **Value**
Change periods (τ)	10, 25, 50, 100, 200	$\lceil n \cdot s \rceil, s = \{1, 2.5, 5, \ldots, 22.5, 25.0\}$
Change magnitudes (ρ)	0.1, 0.25, 0.5, 0.75, 1.0	0.1, 0.2, ..., 1.0
Number of independent runs (R)	50	30
Elitism criteria	TRUE	FALSE
Stopping criterion (B)	1000 iterations	$\tau * 100$ (99 changes)
"Population" size	n ($\lceil 0.25n \rceil$ ants for ACO)	25
Immigrant replacement rate	0.2	0.5 (EIACO)
RIGA mutation probability	0.01	-
RIPBIL learning rate	0.25	-
RIPBIL mutation probability	0.02	-
RIPBIL shift operator	0.05	-
RIACO initial pheromone trail	$1/n$	$1/n$
RIACO relative influence rate	$\alpha = 1, \beta = 5$	$\alpha = 1, \beta = 5$
Pheromone evaporation rate	-	$\theta = 0.8$ (MMAS & MC-MMAS)
Memory size	$\lceil 0.25n \rceil$	3 (PACO & EIACO)
Performance metrics	F_{BOG} & $H_{\Delta m}$	F_{BOG} & $P_{\Delta m}$ & $\mathcal{R}$

5. Experimentation and Results

In this section, we conduct an elusivity analysis on the extended experimental frameworks in [2, 22, 23, 37, 42] to (i) introduce the elusivity concept and accurately identify the elusivity portrait of problems to algorithms, (ii) characterise changes as *elusive* or *non-elusive* to well-studied algorithms, (iii) analyse the adaptative advantage and the behaviour of state-of-the-art online algorithms regarding the elusivity of a realistic problem, and (iv) compare the real algorithms' performance between them. That said, we divide the experimentation in two parts.

First, we replicate and evaluate the elusivity concept by a comprehensive and straightforward case study, where the landscape is rotated for the travelling salesman problem (DTSP) and the knapsack problem (DKP) [2, 37, 42]. In the case of DTSP, the instances[4] `kroA100`, `kroA150` and `kroA200` (containing 100, 150 and 200 cities) are used to insert dynamism, and their best-known values are obtained from [47], used for the accuracy measure. For

[4]Available at http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

DKPs, three static instances have been constructed following the patterns presented in [22], and they are available online[5] repository as supplementary material. The parameter setting employed for this experimentation (extracted from previous works [2, 22, 37, 42]) is described in the second column (Case Study I) of Table 2.

In the second case study, our goal is to exploit the elusivity concept to analyse the advantage of adapting over restart and compare the "real" performance of algorithms in a more sophisticated and realistic framework, and compare to the conclusions drawn in the original experiments. That said, we reproduce (run available code to obtain the same results) and extend the experimentation in [23] to see the entire elusivity portrait of problems to four different ant colony variants with the same parameters employed. The authors insert dynamism (traffic factor) to the instances[4] `kroA100` and `kroA200` (with 100 and 200 cities, respectively) considering 110 change configurations (10 change magnitudes and 11 change frequencies, respectively). The parameters used in our experimentation are described in the third column (Case Study II) of Table 2.

Note that, due to the space limit, only certain results that illustrate the elusivity concept are selected and presented in the following sections. Complete implementation and results are available online[5].

5.1. Case Study 1: Introductory Elusivity Analysis

For illustrative purposes, we select a set of four problems with different dynamism, two algorithms and two performance metrics as example. We denote as P_1 and P_2 two DTSPs that change every $\tau = 100$ iterations with magnitudes $\rho = 0.1$ and $\rho = 0.5$, respectively. Similarly, P_3 and P_4 represent two DKPs changing at period $\tau = 100$ and magnitudes $\rho = 0.1$ & $\rho = 0.5$. Note that P_1 & P_2 are minimisation problems, and P_3 & P_4 are maximisation problems. As for the algorithms and performance metrics, RIPBIL and RIGA are employed under the best-of-generation and the accuracy measures. Table 3 summarises the overall performances of the algorithms, and also captures the elusivity of problems to the algorithm and performance metric combinations (following Definition 1). The results show that online algorithms are useful on P_1–P_3, but restarting the RIGA is beneficial on P_4. In other words, P_4 proves *elusive* to RIGA, regardless of the performance

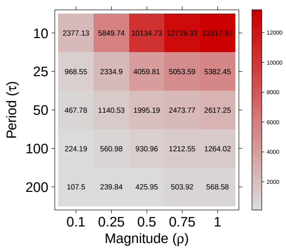
[5]Available at https://zenodo.org/record/7346818#.Y3yfwOx_pqs

Table 3: Overall performances of the algorithms and, in bold, the elusivity value for each problem, algorithm and performance metric combination (elusive problems highlighted in orange).

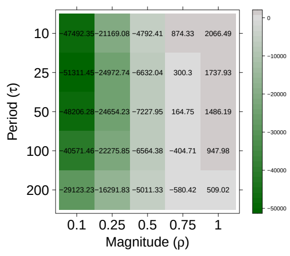
	$\mathbb{E}[\boldsymbol{F_{BOG}}]$	$\mathcal{E}(P, A, \phi)$	$\mathbb{E}[\boldsymbol{H_{\Delta m}}]$	$\mathcal{E}(P, A, \phi)$
$(P_1, RIPBIL)$	186049.80	**-31204.70**	170126.30	**-34361.50**
$(P_1, RIPBIL^r)$	217254.50		204487.80	
$(P_2, RIPBIL)$	208870.90	**-8847.90**	191811.10	**-13238.90**
$(P_2, RIPBIL^r)$	217718.80		205050.00	
$(P_3, RIGA)$	1788.26	**-9.03**	20.40	**-9.00**
$(P_3, RIGA^r)$	1779.23		29.40	
$(P_4, RIGA)$	1773.59	**5.95**	31.00	**2.00**
$(P_4, RIGA^r)$	1779.54		29.00	

metric.

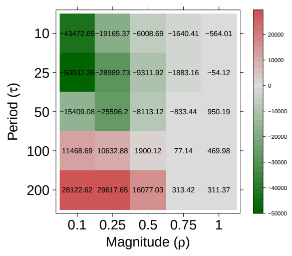
Once the elusivity concept has been introduced and described, we proceed to replicate the first case study. Several change period and magnitude combinations are considered to change the same initial problem, and the elusivity values are shown in a two-dimensional representation (heatmaps) based on the period and magnitude combinations, as shown in Figure 1. Each cell of the heatmap represents the elusivity of the problem, with a specific period-magnitude setting, to an algorithm. Figures 2, 3 and 4 display in heatmaps the elusivity of DTSPs and DKPs with the landscape rotation, solved by RIACO, EIGA, RIGA and RIPBIL and measured by the best-of-generation measure, $F_{BOG}(P, A)$. The colours in the tables are used for guidance only, where the red colour indicates the problem is elusive to the algorithm; also, the higher its intensity, the larger elusivity of the problem to the algorithm. On the contrary, the more green the colour, the less elusive the problem is to the algorithm.



(a) RIACO

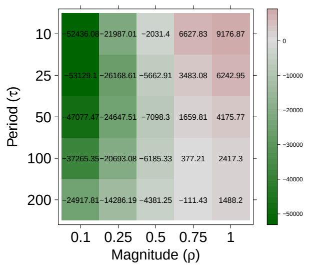


(b) RIGA

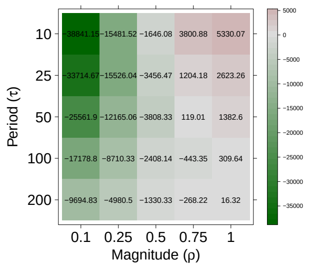


(c) RIPBIL

Figure 2: Elusivity heatmaps of DTSPs with landscape rotation constructed from `kroA150` to RIACO, RIGA and RIPBIL under the best-of-generation.

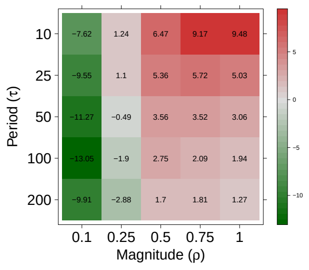


(a) Best-of-generation

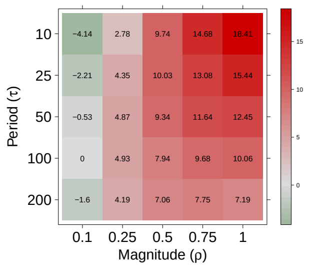


(b) Accuracy

Figure 3: Effect of the performance metric on DTSPs with landscape rotation constructed from `kroA150` to EIGA under best-of-generation and accuracy.



(a) RIGA



(b) RIPBIL

Figure 4: Elusivity heatmaps of DKPs to RIGA and RIPBIL under the best-of-generation.

We can extract the following observations from the heatmaps. First, they demonstrate that adapting to slightly changing problems (left side) is generally beneficial rather than restarting, although the preference is gradually reversed to the point that restarting after a change is more favourable to severely changing problems (right side). This result may sound obvious, since slight changes retain certain similarities of the previous problem instance, and the increase on the magnitude and frequency of changes (decrease of the period) complicates the algorithm adaptability. However, obtained results show that this statement may not hold for certain cases. For example, DTSPs with landscape rotation changing at $\rho = 0.1$ and $\tau = 200$ (see Figure 2(c)) are more elusive to RIPBIL under F_{BOG} than the same problem changing at $\rho = 0.75$ and $\tau = 200$; in other words, the adaptative advantage of RIPBIL is lower in DTSPs with landscape rotation changing slightly and occasionally than the same problem changing severely and occasionally under the best-of-generation metric. This is due to the nature of the random immigrants-based approach, where the new immigrants may not be useful for reacting to slight changes, and since the period of change is large, restarting is more convenient.

Second, the images show different elusivity values for every problem and algorithm combination. For example, DTSPs with landscape rotation changing at $\rho = 1$ and $\tau = 200$ are less elusive to RIACO under F_{BOG} than the same problem changing at $\rho = 1$ and $\tau = 10$ (see Figure 2(a)); in other words, the adaptative advantage of RIACO is higher in DTSPs changing severely and occasionally than the same problem changing severely and frequently. Therefore, we can say that the same algorithm proves more or less *elusive* depending on the problem configuration.

Third, from Figure 3, we can see different elusive values for DTSPs with landscape rotation to EIGA under both performance metrics, although the general pattern of elusivity holds for both performance metrics. That is, for example, DTSPs that change (rotate) at $\rho \leq 0.5$ are non-elusive to EIGA under F_{BOG} and $H_{\Delta m}$ to different degrees, depending on the frequency and magnitude of the change, and the performance metric considered. Therefore, these observations highlight the effect of the performance metric on the elusivity formulation.

Fourth, by contrasting Figures 2(b) and 3(a) we can analyse the influence of the adaptation mechanism on the elusivity of DTSPs with landscape rotation to EIGA and RIGA under best-of-generation. Based on this comparison, we can observe slightly different elusivity values of the DTSPs for

EIGA and RIGA under F_{BOG}, respectively, although the general elusivity pattern is certainly similar for both immigrant-based algorithms. That is, although to different degrees, slightly and frequently changing DTSPs are non-elusive to EIGA and RIGA under F_{BOG}, and severely and frequently changing DTSPs are elusive under the same conditions.

Finally, the results show that landscape rotation generates mostly *non-elusive* DTSPs no matter the algorithm used, i.e. adapting to changes is usually beneficial (only 40 problems out of 125 are *elusive* according to Figures 2 and 3). In the case of the DKPs, however, online algorithms are rarely beneficial, since 38 problems out of 50 prove *elusive* to random immigrants-based algorithms (see Figure 4). Hence, the inclusion of elusivity analysis adds value by revealing that, under the set conditions, restart is more effective than random immigrant adaptation on DKPs generated by the XOR DOP benchmark generator. Moreover, it is worth noting that DKPs changing at $\rho = 0.1$ and $\tau = 100$ are on the threshold to prove elusive to RIPBIL under F_{BOG}. Hence, in order to validate the experimental results, we will perform a statistical analysis in the next section.

5.1.1. Statistical Analysis

In order to ensure that such results are still valid when assessing the uncertainty related to the experimentation, a Bayesian statistical analysis, equivalent of the pairwise Wilcoxon signed-rank test[6], is carried out. This technique estimates the expected probability of two algorithms obtaining the best results (winning probability), given some observations (experimental results), and some prior belief (usually uniformly generated from a Dirichlet distribution [49, 50]). We consider that the two algorithms to be compared perform similarly (tying probability) if the performance difference between them is within the Range Of Practical Equivalence (ROPE):

$$ROPE = (0, |\,\mathbb{E}[\phi(A^r, P)]| \cdot \gamma), \tag{8}$$

where $\mathbb{E}[\phi(A^r, P)]$ is the mean performance of the restarting version of the algorithm A on the problem P under the performance metric ϕ and γ is a variance parameter. Certainly, the γ parameter is used to define the ROPE of the contrasted algorithms in the Bayesian analysis, where a value of $\gamma = 0$ denotes that two algorithms have equivalent performance when the difference

[6]Available in the R package `scmamp` [48].

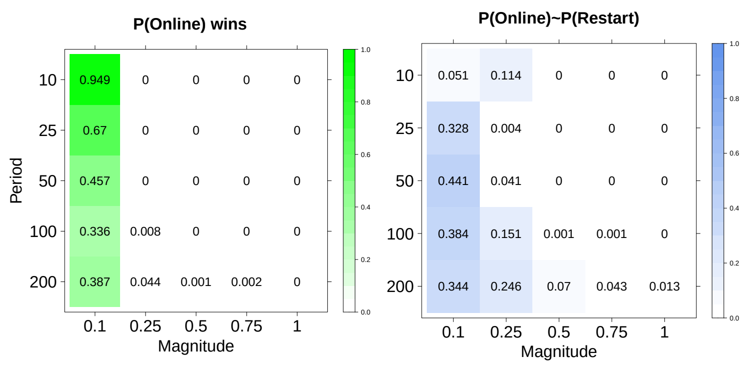


Figure 5: Elusivity heatmaps showing winning probabilities of online RIPBIL against RIPBIL^r (in green) and equivalent performance probabilities (in blue) when solving DKPs with landscape rotation and measured by best-of-generation.

in scores is equal to 0. We set $\gamma = 0.001$ to be a relatively low value to consider both algorithms performing similarly. Note that this strategy could be also added to the elusivity formulation by replacing the zero in Definition 2 with $|\mathbb{E}[\phi(A^r, P)]| \cdot \gamma$.

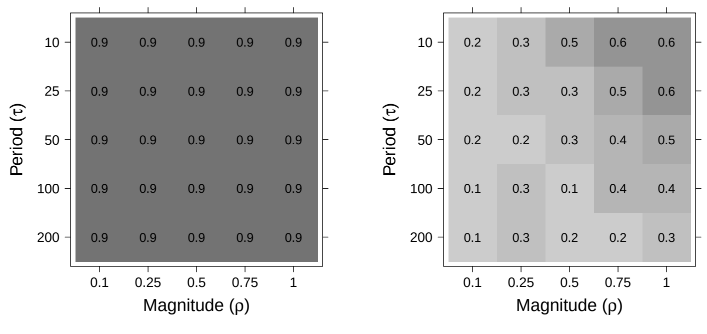
Based on this analysis, we wish to extend the previously shown elusivity heatmaps to more general *statistical heatmaps*. Particularly, for each experimental setting, two complementary heatmaps are created: (i) the first shows the probability of the online algorithm being the winner (green cells), and (ii) the second shows the probability of the equivalent performance (blue cells) of both algorithm.

To illustrate the statistical analysis of the elusivity of the results obtained in the previous section, we consider the combination of DKPs with landscape rotation, RIPBIL and best-of-generation (see Figure 5). Guided principally by the colours, the winning heatmap (green) in Figure 5 looks similar to the elusivity heatmaps presented in Figure 4(b). Nevertheless, the heatmaps in Figure 5 demonstrate that most DKPs with landscape rotation prove elusive to RIPBIL in this experimental setting, although for some problems, RIPBIL and RIPBIL^r achieve similar performance. Only two DKPs changing at $\rho = 0.1$ and $\tau = 10, 20$ prove clearly *non-elusive* to RIPBIL under F_{BOG}, whereas problems changing at $\tau = 50, 100, 200$ and $\rho = 0.1$ are in the threshold to prove elusive to RIPBIL under F_{BOG}. That is, the winning and tying probabilities of RIPBIL over RIPBIL^r are practically equivalent for DKP that change in $\tau = 50, 100, 200$ and $\rho = 0.1$. On the contrary, restarting is better, or no worse, than adapting to DKPs changing at $\rho \geq 0.25$, no matter the period between changes. Hence, we can say that this configuration is narrowly *elusive* to RIPBIL under best-of-generation. The rest of DKP configurations are *elusive* to RIPBIL. Hence, we can deduce that the XOR DOP generator constructs mostly *elusive* DKPs for RIPBIL under best-of-generation.

5.1.2. Effect of the Parameter Setting on the Elusivity

As previously mentioned, the immigrant replacement rate determines the number of immigrants that substitute the less promising (worst) solutions of the population of immigrants-based algorithms at every iteration. In previous experiments, the replacement ratio was set to 0.2, a typical value used in the literature [37, 38, 51]. Hence, in order to analyse the influence of this parameter on the performance of immigrant-based algorithms, the immigrant replacement rate is set to $\{0.1, 0.2, \ldots, 0.9\}$.

First, we have performed an evaluation of the performance achieved by the immigrants-based online algorithms (for each problem under the considered performance metrics) to get the optimal immigrant replacement rates. Figure 6 shows in heatmaps the optimal immigrant replacement rate setting for EIGA and RIGA to solve DTSPs with different frequency and magnitude



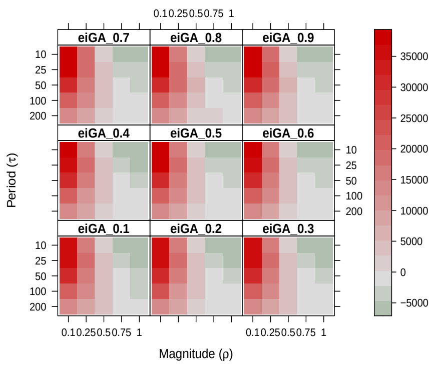
(a) EIGA (b) RIGA

Figure 6: Optimal immigrant replacement rate of EIGA and RIGA under best-of-generation for DTSPs with landscape rotation constructed from `kroA100`.

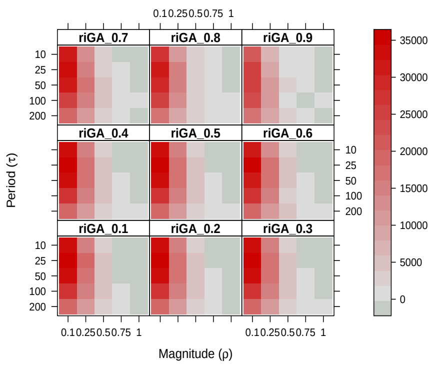
of change under best-of-generation, respectively.

Figure 6(a) shows that the largest immigrant replacement rate for EIGA obtains the best performance under F_{BOG} when solving DTSPs with landscape rotation, no matter the frequency and magnitude of change. Furthermore, smaller replacement rates for RIGA generally achieve better performance under F_{BOG} (see Figure 6(b)), although it varies by the frequency and magnitude of changes on DTSPs. Specifically, smaller replacement rates are preferred for occasional and slightly changing problems, and gradually, it is increased together with the frequency and magnitude of change, to the point that larger replacement rates are more suitable for frequent and severe changes. These observations are consistent with the observations found in literature [38].

Afterwards, once optimal parameter settings are calculated for each problem and algorithm, we have performed an analysis to study the influence of the immigrant replacement rate on the elusivity of problems to the algorithms (for each parameter setting) under the performance metrics considered. Figure 7 shows a set of heatmaps that reveal the elusivity values obtained for each EIGA and RIGA with a defined parameter setting for each DTSP under best-of-generation, respectively. As can be seen in the figure, the elusivity



(a) EIGA



(b) RIGA

Figure 7: Elusivity heatmaps showing the influence of immigrant replacement rates for EIGA and RIGA, under best-of-generation, on the elusivity of DTSPs with landscape rotation constructed from `kroA100`.

of DTSPs with landscape rotation to EIGA and RIGA is roughly sensitive (under best-of-generation metric) to the considered algorithmic parameters, although the general elusivity pattern is maintained. That is, although the elusive values vary slightly with the modification in the immigrant replacement rate, any setting of this parameter would illustrate the applicability of the concept of elusivity. Hence, even if we apply the optimal replacement rate on the immigrants-based algorithms for each dynamic problem (from Figure 6), some DTSPs would still prove elusive to EIGA under F_{BOG} (for example, see the top-right heatmap in Figure 7(a)).

5.1.3. Overall outcome

Although the above statements ratify the utility of the elusivity formulation, they do not quantify the extent to which elusivity can feature in published work. Therefore, we compute the expected probability of the landscape rotation creating *elusive* problems (for the considered algorithms and performance measures) as follows. First, the expected performance of the online and restart algorithms are calculated. Then, the elusivity (following Definition 1) of each problem to the algorithms and performance metrics is quantified. Finally, after performing a Bayesian analysis, we count and average the highest posterior probability for each problem (with all combinations of frequency and magnitude of change) being elusive to the algorithms under performance metrics considered.

For clarification purposes, let's consider Figure 5 as an example. We can observe that online RIGA is superior 4 times out of 25, and ties restart once out of 25. Hence, these results reflect an expected probability of 0.12 to produce non-elusive DTSPs with landscape rotation to RIGA under the best-of-generation, and a probability of 0.08 to produce DTSPs with landscape rotation where online and restart RIGA are practically equivalent.

That said, the expected probability that permutation-based landscape rotation creates *elusive* DTSPs is 0.2. In the case of the XOR DOP, the expected probability is 0.66 that the generated problems are *elusive*, and 0.14 represents practical equivalence.

Now that we can ensure the generation of dynamic, but non-elusive, problems to algorithms, we focus on extending the elusivity concept to quantify the effectiveness of adaptation mechanisms or comparing algorithm performances regarding the elusivity.

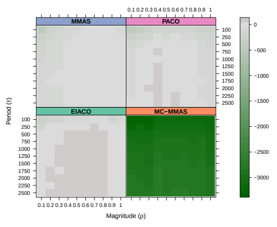
5.2. Case Study 2: Advanced Elusivity Analysis

In the previous section, we demonstrated that the elusivity of problems generated by the landscape rotation, which are mainly used for academic purposes, varies with the change period and magnitude tuning, as well as with the selected algorithm. In this section, the goal is to make the most of the elusivity formulation to capture and evaluate the adaptative advantage and behaviour of online algorithms on a more realistic framework. The idea is to use the elusivity concept to measure the extent to which adaptation improves or degrades algorithm performance in comparison to restart.

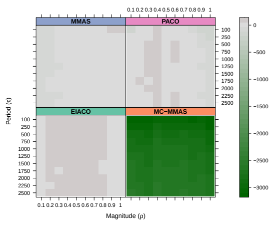
The experiments carried out are reproduced from a state-of-the-art research work [23], where DTSPs with different dynamisms (traffic factor and city replacement), four ant colony variants (MMAS, PACO, EIACO and MC-MMAS) and three performance metrics are used. The authors categorise changes as *fast* when the change period is $\tau \leq 2.5n$, and *slow* for $\tau \geq 25n$. Similarly, they refer a change to be *small*, *medium* or *large* based on the magnitudes $\rho = 0.1, 0.25, 0.5, 0.75$. However, we can observe that change magnitudes are unevenly distributed across categories, as four values ($\rho = 0.1, 0.25, 0.5, 0.75$) are assigned to three categories (*small*, *medium* or *large*).

The paper states that adaptation mechanisms enhance the adaptability of algorithms, although their performance depends on the settings (dynamism) of the problem. Besides, they exhibit the following observations. First, PACO and EIACO outperform MMAS and MC-MMAS for most quickly changing DTSPs under F_{BOG}. Second, MMAS and MC-MMAS perform better than PACO and EIACO for most slowly changing DTSPs under F_{BOG} and $P_{\Delta m}$, although it is gradually inverted with the increase of problem size. Third, all algorithms obtain very good results under $\mathcal{R}$. Fourth, the restarting version of the algorithms are not effective for DTSPs with traffic factor when changes do not affect the best solution found, since it would result in an undetected change for the restarting algorithm.

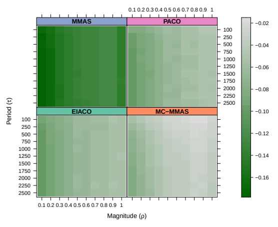
Finally, authors assure that the following statements are consistent with the observations found in their previous studies [26]: (i) MC-MMAS is competitive with MMAS, i.e. both maintain a competitive performance; (ii) EIACO generally outperforms PACO; and (iii) PACO gradually outperforms MMAS as problem size increases.



(a) Best-of-generation

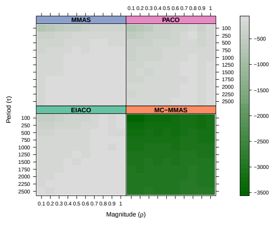


(b) Best-before-change

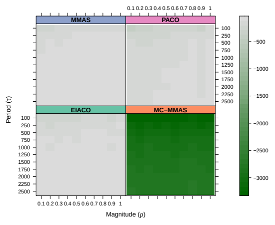


(c) Robustness

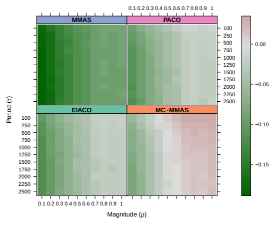
Figure 8: Elusivity heatmaps of DTSPs with city replacement to four algorithms under three different performance metrics.



(a) Best-of-generation

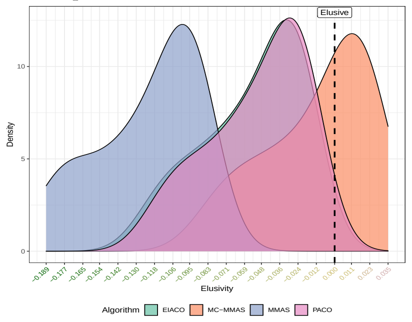


(b) Best-before-change

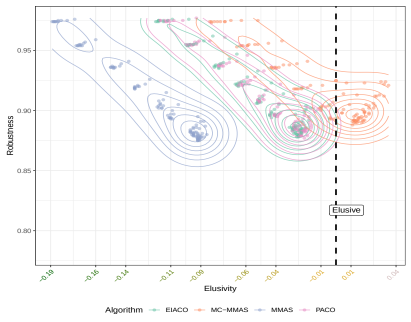


(c) Robustness

Figure 9: Elusivity heatmaps of DTSPs with traffic factor to four algorithms under three different performance metrics.



(a) Density plot



(b) Scatter plot

Figure 10: Elusivity as a measure for adaptative advantage algorithms on DTSPs with traffic factor under robustness. The elusivity threshold (zero) is represented by a vertical dashed line, and the algorithms by the colour of the header of the heatmaps.

5.2.1. Elusivity Analysis

First of all, note that the aim of this experimentation is to illustrate and study the application of the elusivity concept to quantify the adaptability of online algorithms over restarting the search after detecting a change. To that end, we have decided to extend the change period and magnitude settings used in [23], and show the elusivity values in heatmaps to accurately highlight the adaptative advantage of online algorithms to solve DTSPs with different settings. Figures 8 and 9 capture the elusivity of the DTSPs with city replacement and traffic factor, respectively, to the algorithms under three performance metrics.

Figures 8(a), 8(b), 9(a) and 9(b) demonstrate that most DTSPs (either for city replacement or traffic factor) prove *non-elusive* to the algorithms under F_{BOG} and $P_{\Delta m}$. MC-MMAS exhibits lower elusivity than PACO, MMAS and EIACO under F_{BOG} and $P_{\Delta m}$, meaning that the adaptative advantage over its restart version is large under these performance metrics. The heatmaps for PACO, MMAS and EIACO reveal that adaptation is slightly better than restarting the search for slight and frequent changing DTSPs, although both online and restarting versions obtain similar results in general. Hence, DTSPs with city replacement or traffic factor are in the threshold to become elusive to PACO, MMAS and EIACO under F_{BOG} and $P_{\Delta m}$, respectively, since the adaptation mechanisms for these algorithms confer little or no advantage over their restart for occasional and severe changes.

In the same way, Figure 8(c) shows that DTSPs with city replacement prove non-elusive to the algorithms under $\mathcal{R}$. However, Figure 9(c) shows that some DTSPs with traffic factor that prove elusive to MC-MMAS under $\mathcal{R}$. That is, the robustness of MC-MMAS deteriorates with the increase of the traffic factor, to the point that DTSPs with medium to severe traffic factor become elusive to MC-MMAS under $\mathcal{R}$. This particular case allows us to demonstrate the role of performance metrics in adaptability, apart from the problem and the algorithm. Nevertheless, it is worth noting that most empirical studies in dynamic optimisation design and compare algorithms that aim to optimise the best-of-generation. In fact, as stated in [52], robustness and best-of-generation metrics conflict with each other, where algorithms perform better on problems that prove less robust.

5.2.2. Elusivity as a Measure for the Advantage of Adaptation Mechanisms

So far, we have presented and analysed our results using elusivity heatmaps to quantify the advantage that adaptation brings over a restart of the algorithm. Nevertheless, heatmaps fall short when analysing the effectiveness and sensitivity of the algorithms, and comparing the best performance of each version of algorithms against other algorithms'. Therefore, we extend the elusivity analysis using density plots and scatter plots.

For illustration purposes, we consider the aforementioned paradigmatic case study to analyse the relation between the robustness and elusivity of the four ACO algorithms for DTSPs with traffic factor in more depth. It is worth noting that the observations drawn in the following analysis may not hold for other problem, algorithm and performance metric combination. In fact, as mentioned in [52], the observations drawn from robustness may conflict with the ones from best-of-generation, thus affecting the elusivity analysis. In Figure 10, we support the elusivity heatmap in Figure 9(c) with a density plot and a scatter plot. The density plot in Figure 10(a) uses a kernel probability density to estimate the elusivity distribution for DTSPs with traffic factor to each algorithm under $\mathcal{R}$, described in Equation 7, on the same set of DTSPs (see Table 2). The scatter plot in Figure 10(b) shows the relation between the robustness obtained by the best algorithm version (online or restart) for each problem setting and the elusivity obtained by each algorithm.

Figure 10(a) reveals that PACO, MMAS and EIACO generally prove *non-elusive* under $\mathcal{R}$, whereas the performance of MC-MMAS varies with the frequency and magnitude of change. That is, from the heatmaps, we can observe that DTSPs with traffic factor changing at $\rho = 1$ and $\tau = 100$ prove *elusive* to MC-MMAS under $\mathcal{R}$, whereas problems changing at $\rho = 0.1$ and $\tau = 100$ prove non-elusive. Therefore, we can say that, for these change configurations, the adaptation mechanisms for MC-MMAS is giving little or no advantage in terms of robustness.

Similarly, densities also demonstrate that PACO, MMAS and EIACO have a more concentrated elusivity distribution, whereas the elusivity of DTSPs with traffic factor to MC-MMAS under $\mathcal{R}$ is more variable. This might be because of (i) the high variability of adaptation mechanisms, that stand out depending on the period and frequency setting of changes; or (ii) a bad tuning of adaptation mechanism parameters, such as immigrant replacement rate or the memory size in EIACO or PACO, for example.

Figure 10(b) displays the relation between the robustness of the best version for each algorithm and their respective elusivity in a scatter plot. Points on the left-side (elusivity lower than zero) represent a better performance of online algorithms over their restarting version (PACO, MMAS, EIACO and

MC-MMAS), and the opposite for right-sided points (PACO^r, MMAS^r and EIACO^r). From the plot, we can say that DTSPs with traffic factor prove *non-elusive* to PACO, MMAS and EIACO under $\mathcal{R}$, and also for problems changing at $\rho \leq 0.4$ to MC-MMAS. The figure also shows the line of best fit for each algorithm, aiming to highlight the elusivity and performance trend of algorithms with respect to the configuration of the changes. In a certain way, this chart shows the elusivity distributions in Figure 10(a) from a top view, and the depth is determined by the robustness of the algorithms. Recall that robustness is a maximisation measure, so larger robustness means a better performance of the algorithm.

From the figure, we can observe a similar pattern for all algorithms when measured under robustness, where slightly changing DTSPs are concentrated in the upper part of the plot and severe changes at the bottom, although all algorithms differ in the elusivity distribution. Aforementioned, PACO and EIACO usually prove *non-elusive* for DTSPs with traffic factor under $\mathcal{R}$, although points are concentrated close to the threshold to become *elusive* (elusivity close to zero) as the frequency and magnitude of change increase. MC-MMAS^r proves more or less *elusive* depending on the setting of changes under $\mathcal{R}$, although its robustness is certainly maintained, no matter the frequency and magnitude of change. That said, we can conclude that, in this case study, the adaptation mechanisms for MC-MMAS may be disadvantageous for some frequencies and magnitudes of change when measured under robustness, although the algorithm is quite robust by nature. Nevertheless, note that these observations are drawn from a particular illustration of the obtained results, and they are limited to the exposed experimental setup. Finally, it is interesting to note the trend on the distribution of the algorithms: the less elusive, the more robust become the algorithms. Therefore, we can conclude that the adaptative advantage is influenced by the change frequency and magnitude setting.

From these observations, we can state that PACO, MMAS and EIACO are less robust than MC-MMAS^r for some change settings, i.e. except from the DTSPs changing at $\rho \geq 0.5$, even a restarting behaviour of MC-MMAS can be more effective than PACO, MMAS and EIACO under $\mathcal{R}$. So we demonstrate that adaptation mechanisms do not always improve the robustness of algorithms, since PACO, EIACO and MMAS are less robust than MC-MMAS^r for DTSPs with traffic factor changing at $\rho \geq 0.5$. Obtained results suggest the inclusion of the restart, in future research, to avoid erroneous evaluation of algorithms in elusive problems where there is no adaptative advantage.

Finally, an interesting observation that we cannot ignore is that, for each algorithm, points are grouped based primarily on the magnitude of changes, but also on the change period. This statement allows us to display, in a certain way, the characterisation of problems in regard to the performance and elusivity of algorithms. For example, from Figure 10(b), we can discern 7 different groups for PACO, MMAS and EIACO, and 6 for MC-MMAS^r. However, the characterisation of the groups varies with the algorithm version. For MMAS, EIACO and PACO, the increase in the change magnitude is related to the variability in robustness and the concentration of elusivity of groups. In the case of MC-MMAS^r, the robustness is maintained for each period of change (the magnitude does not influence the performance of restarting algorithms), while the elusivity of the groups varies with the magnitude of the change, i.e. the more severe the change, the more effective is the restart over the adaptation for MC-MMAS.

5.2.3. Overall outcome

This section has demonstrated that the elusivity formulation can be extended to evaluate the adaptation mechanism over a well-defined behaviour of algorithms. The different visual representations have revealed that (i) heatmaps are useful to get the elusivity portrait of problems (or benchmark generators) varying their change frequency and magnitudes, (ii) density plots give us an insight of the elusivity and the adaptative advantage of algorithms on a problem set, and (iii) scatter plots allow us to compare the best performance of the algorithms.

6. Conclusions and Future Work

The field of dynamic optimisation presents a wide variety of algorithms with adaptation mechanisms to solve problems that change over time, either adapting or reacting to problem changes. However, empirical works usually compare the performance of algorithms under the frequency and magnitude of changes, although they often ignore whether the problem is amenable to be solved by an online algorithm. This work has validated that the frequency and magnitude of changes alone are insufficient to determine the difficulty of dynamic problems to algorithms that adapt to changes, since it also depends on the repercussion of problem variations on algorithms' performance. To that end, the elusivity concept has been introduced and evaluated.

The mathematical formulations and the systematic analysis of the elusivity concept proved the validity to (i) distinguish *elusive* and *non-elusive* problems based on the adaptability of algorithms to changes, and (ii) evaluate the advantage of the adaptation mechanisms over the restart. This study suggests avoiding algorithm performance comparison under change classifications based only on the frequency and magnitude of changes, since the dynamism also depends on the adaptation challenge of algorithms to deal with changes and the performance metric used. In fact, performance metrics are usually neglected in previous research on this topic, but they are crucial to a full definition of the elusivity.

The conducted experiments demonstrated the existence of elusive problems in already published studies [2, 22, 23, 37, 42] to different extents according to the problem, algorithm and performance metric combination. Therefore, this work suggests including the restarting version of the algorithms in the experimentation to eliminate erroneous study of algorithms with disadvantageous adaptation mechanisms in future research. Note that our definitions do not make any assumptions about the problem, algorithm, performance metric, representation, type of dynamism, etc. so it can be applied into any dynamic optimisation research.

There are many studies that emerge from this work. The first and most obvious is the extension of this preliminary work to explore other properties of benchmark generators and their elusivity values for particular algorithms. It may be interesting to analyse the elusivity of a real-world applications for different dynamics that may happen. For example, extend this study to the consideration of dynamic problems with dimensional changes. This way, we should guarantee the generation of *non-elusive* dimensionally varying DTSPs to algorithms.

The other way round, the elusivity may be also extended to considering changes that are impossible or sensitive to adapt (robust optimisation) [16, 53]. In such cases, the goal is to find a sequence of solutions that preserve an acceptable quality during a time interval until it deteriorates.

Finally, an interesting and promising current is to predict the elusivity of a dynamic problem to an algorithm. In a real world situation, it can be useful to quantify the elusivity of a problem in advance (offline), ideally in toy experimentation, before running the online or the restarting algorithm version. This idea can also be extended to develop a predictive adaptation mechanism capable of deciding online when to restart, adapt or maintain the algorithm search based on the information collected.

Acknowledgements

We wish to thank Michalis Mavrovouniotis, Shengxiang Yang, Mien Van, Changhe Li, and Marios Polycarpou for letting us access their code for reproduction purposes.

Funding: This work has been partially supported by the Research Groups 2022-2024 (IT504-22), the Elkartek Program (KK-2022/00106, SIGZE, KK-2021/00065) from the Basque Government, and the PID2019-106453GA-I00/AEI/10.13039/501100011033 research project from the Spanish Ministry of Science.

References

[1] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proc. on CEC, 1999, pp. 1875–1882.

[2] S. Yang, X. Yao, Dual population-based incremental learning for problem optimization in dynamic environments, in: Asia Pacific Symposium on Intelligent and Evolutionary Systems, 2003, pp. 49–56.

[3] A. Younes, P. Calamai, O. Basir, Generalized benchmark generation for dynamic combinatorial problems, in: Proc. on GECCO, 2005, pp. 25–31.

[4] M. Mavrovouniotis, S. Yang, X. Yao, A benchmark generator for dynamic permutation-encoded problems, in: Proc. on PPSN, 2012, pp. 508–517.

[5] T. T. Nguyen, Continuous dynamic optimisation using evolutionary algorithms, Ph.D. thesis, University of Birmingham, 2011.

[6] P. Rohlfshagen, P. K. Lehre, X. Yao, Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change, in: Proc. on GECCO, 2009, pp. 1713–1720.

[7] P. A. N. Bosman, Learning, anticipation and time-deception in evolutionary online dynamic optimization, in: Proc. on GECCO, 2005, pp. 39–47.

[8] S. Yang, Y. Jiang, T. T. Nguyen, Metaheuristics for dynamic combinatorial optimization problems, in: IMA Journal of Management Mathematics, 2012, pp. 451–480.

[9] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, in: Swarm and Evolutionary Computation, 2012, pp. 1 – 24.

[10] J. Branke, Evolutionary optimization in dynamic environments, volume 3, Springer Science & Business Media, USA, 2002.

[11] J. Branke, E. Salihoğlu, c. Uyar, Towards an analysis of dynamic environments, in: Proc. on GECCO, 2005, pp. 1433–1440.

[12] B. Doerr, C. Doerr, F. Neumann, Fast re-optimization via structural diversity, in: Proc. on GECCO, 2019, pp. 233–241.

[13] J. Bossek, F. Neumann, P. Peng, D. Sudholt, Time complexity analysis of randomized search heuristics for the dynamic graph coloring problem, in: Algorithmica, 10, Springer-Verlag, Berlin, Heidelberg, 2021, pp. 3148–3179.

[14] A. Lissovoi, C. Witt, Runtime analysis of ant colony optimization on dynamic shortest path problems, in: Proc. on GECCO, 2013, pp. 1605–1612.

[15] M. Pourhassan, W. Gao, F. Neumann, Maintaining 2-approximations for the dynamic vertex cover problem using evolutionary algorithms, in: Proc. on GECCO, 2015, pp. 903–910.

[16] X. Yu, Y. Jin, K. Tang, X. Yao, Robust optimization over time — a new perspective on dynamic optimization problems, in: Proc. on CEC, 2010, pp. 1–6.

[17] R. Tinós, D. Whitley, A. Howe, Use of explicit memory in the dynamic travelling salesman problem, in: Proc. on GECCO, 2014, pp. 999–1006.

[18] R. Allmendinger, J. Knowles, Evolutionary optimization on problems subject to changes of variables, in: Proc. on PPSN, 2010, pp. 151–160.

[19] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, X. Yao, A survey of evolutionary continuous dynamic optimization over two decades—part b, in: IEEE TEVC, 2021, pp. 630–650.

[20] R. Tinós, S. Yang, Analysis of fitness landscape modifications in evolutionary dynamic optimization, in: Information Sciences, 2014, pp. 214 – 236.

[21] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic tsp, in: Applications of Evolutionary Computing, 2001, pp. 213–222.

[22] S. Yang, X. Yao, Experimental study on population-based incremental learning algorithms for dynamic optimization problems, in: Soft Computing, 2005, pp. 815–834.

[23] M. Mavrovouniotis, S. Yang, M. Van, C. Li, M. Polycarpou, Ant colony optimization algorithms for dynamic optimization: A case study of the dynamic travelling salesperson problem, in: IEEE CIM, 2020, pp. 52–63.

[24] K. Weicker, An analysis of dynamic severity and population size, in: Proc. on PPSN, 2000, pp. 159–168.

[25] P. Rohlfshagen, X. Yao, Attributes of dynamic combinatorial optimisation, in: Simulated Evolution and Learning, 2008, pp. 442–451.

[26] M. Mavrovouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: Algorithms and applications, in: Swarm and Evolutionary Computation, 2017, pp. 1 – 17.

[27] H. G. Cobb, An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, Technical Report, 1990.

[28] T. Bäck, On the behavior of evolutionary algorithms in dynamic environments, in: Proc. on CEC, 1998, pp. 446–451.

[29] C. Cruz, J. R. González, D. A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, in: Soft Computing, 2011, pp. 1427–1448.

[30] J. G. O. L. Duhain, A. P. Engelbrecht, Towards a more complete classification system for dynamically changing environments, in: Proc. on CEC, 2012, pp. 1–8.

[31] R. C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proc. on CEC, 2001, pp. 94–100.

[32] P. J. Angeline, Tracking extrema in dynamic environments, in: Evolutionary Programming, 1997, pp. 335–345.

[33] K. De Jong, Evolving in a changing world, in: Foundations of Intelligent Systems, 1999, pp. 512–519.

[34] J. Alza, M. Bartlett, J. Ceberio, J. McCall, Towards the landscape rotation as a perturbation strategy on the quadratic assignment problem, in: Proc. on GECCO, 2021, pp. 1405–1413.

[35] J. Alza, M. Bartlett, J. Ceberio, J. McCall, On the definition of dynamic permutation problems under landscape rotation, in: Proc. on GECCO, 2019, pp. 1518–1526.

[36] M. Guntsch, M. Middendorf, H. Schmeck, An ant colony optimization approach to dynamic tsp, in: Proc. on GECCO, 2001, p. 860–867.

[37] S. Yang, Genetic algorithms with memory-and elitism-based immigrants in dynamic environments, in: Evolutionary Computation, 2008, pp. 385–416.

[38] M. Mavrovouniotis, S. Yang, Population-based incremental learning with immigrants schemes in changing environments, in: 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 1444–1451.

[39] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, in: Applied Soft Computing, 2013, pp. 4023 – 4037.

[40] R. Tinós, S. Yang, A self-organizing random immigrants genetic algorithm for dynamic optimization problems, in: Genetic Programming and Evolvable Machines, 2007, pp. 255–286.

[41] J. J. Grefenstette, et al., Genetic algorithms for changing environments, in: Proc. on PPSN, 1992, pp. 137–144.

[42] M. Mavrovouniotis, S. Yang, Elitism-based immigrants for ant colony optimization in dynamic environments: Adapting the replacement rate, in: Proc. on CEC, 2014, pp. 1752–1759.

[43] S. Yang, Memory-enhanced univariate marginal distribution algorithms for dynamic optimization problems, in: Proc. in CEC, 2005, pp. 2560–2567.

[44] K. Trojanowski, Z. Michalewicz, Searching for optima in non-stationary environments, in: Proc. on CEC, 1999, pp. 1843–1850.

[45] C. Li, S. Yang, T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H. Beyer, P. Suganthan, Benchmark generator for CEC 2009 competition on dynamic optimization, Technical Report, 2008.

[46] W. Rand, R. Riolo, Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions, in: Proc. on GECCO, 2005, pp. 32–38.

[47] M. Mavrovouniotis, S. Yang, Empirical study on the effect of population size on max-min ant system in dynamic environments, in: Proc. on CEC, 2016, pp. 853–860.

[48] B. Calvo, G. Santafe, scmamp: Statistical comparison of multiple algorithms in multiple problems, in: The R Journal, 2015, pp. 248–256.

[49] B. Calvo, J. Ceberio, J. A. Lozano, Bayesian inference for algorithm ranking analysis, in: Proc. on GECCO, 2018, pp. 324–325.

[50] B. Calvo, O. M. Shir, J. Ceberio, C. Doerr, H. Wang, T. Bäck, J. A. Lozano, Bayesian performance analysis for black-box optimization benchmarking, in: Proc. on GECCO, 2019, pp. 1789–1797.

[51] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes in dynamic environments, in: Proc. on PPSN, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 371–380.

[52] X. Yu, T. Chen, X. Yao, Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization, Memetic Computing 1 (2009) 3–24. doi:10.1007/s12293-008-0003-6.

[53] Yaochu Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, in: IEEE TEVC, 2005, pp. 303–317.