On the Use of Dynamic GP Fitness Cases in Static and Dynamic Optimisation Problems

Edgar Galván-López[1*], Lucia Vázquez-Mendoza[2], Marc Schoenauer[3] and Leonardo Trujillo[4]

[1] Department of Computer Science, National University of Ireland Maynooth, Ireland
edgar.galvan@mu.ie
[2] School of Social Sciences and Philosophy, Trinity College Dublin, Ireland
lucyvaz@gmail.com
[3] TAU, INRIA and LRI, CNRS & U. Paris-Sud, Université Paris-Saclay, France
marc.schoenauer@inria.fr
[4] Posgrado en Ciencias de la Ingeniería, Instituto Tecnológico de Tijuana, México
leonardo.trujillo@tectijuana.edu.mx

Abstract. In Genetic Programming (GP), the fitness of individuals is normally computed by using a set of fitness cases (FCs). Research on the use of FCs in GP has primarily focused on how to reduce the size of these sets. However, often, only a small set of FCs is available and there is no need to reduce it. In this work, we are interested in using the whole FCs set, but rather than adopting the commonly used GP approach of presenting the entire set of FCs to the system from the beginning of the search, referred as static FCs, we allow the GP system to build it by aggregation over time, named as dynamic FCs, with the hope to make the search more amenable. Moreover, there is no study on the use of FCs in Dynamic Optimisation Problems (DOPs). To this end, we also use the Kendall Tau Distance (KTD) approach, which quantifies pairwise dissimilarities among two lists of fitness values. KTD aims to capture the degree of a change in DOPs and we use this to promote structural diversity. Results on eight symbolic regression functions indicate that both approaches are highly beneficial in GP.

1 Introduction

Normally, the fitness of Genetic Programming (GP) [9] programs is obtained by using a set of fitness cases: a fitness case is an input/output pair and the fitness of an individual is measured on how well it matches the output(s) from input(s).

Research on the use of fitness cases has primarily focused on how to reduce the number of these cases when running a GP system given that this is a major element that affects speed [6, 11, 19, 14].

* Research conducted during Galván's stay at TAU, INRIA and LRI, CNRS & U. Paris-Sud, Université Paris-Saclay, France.

There are, however, some problems where only a few fitness cases are available for the GP system to work with. For instance, when dealing with highly binary unbalanced data for a classification task, the positive (minority) class has only a few cases and the use of all the available fitness cases is necessary [2, 4]. Other times, it may be the case that the dataset is highly contaminated by outliers and sampling is required to detect true examples of the system [12].

In this work, rather than using only a subset of fitness cases from the entire set [6, 11, 19], we are interested in using them all in a way to make the search more robust. To do so, we propose an approach called dynamic fitness cases, wherein cases are built by aggregation over generations instead of using the commonly adopted approach of using them all from the beginning of the search.

Moreover, there is no study that has focused its attention on the study of fitness cases on dynamic optimisation problems (DOPs). These are problems that are solved online by an optimisation algorithm as time progresses [16]. This work uses both static problems and DOPs to test the proposed approach.

Multiple elements have been reported to be beneficial in DOPs (see [16] for a detailed analysis on the area). One key element is diversity. This is a key element of the biological theory of natural selection and it is used in EAs to describe, for instance, structural variety, and it is expected that an EA with a mechanism to promote diversity will greatly improve its performance [16].

Diverse approaches have been proposed to promote diversity in EAs. One commonly adopted approach is the replacement of individuals in a population by newly generated genetic material. However, a common element observed when doing so is that frequently researchers use an arbitrary approach to decide the number of individuals that need to be replaced [13, 17, 21]. However, this process is purely intuitive and often expensive due to its trial-and-error nature.

To address this issue, we also use a mechanism to make a more informed decision to determine the proportion of individuals that need to be replaced in DOPs. To this end, we use the fitness values of individuals as indicators to determine how big/small a change is, and consequently, use this information to determine, for instance, the number of individuals in a population that need to be replaced by new individuals. Any population-based EA can adopt our proposed approach and in this work, as stated previously, we use a GP system.

Thus, the main contributions of this work are: (a) the use of dynamic fitness cases, wherein cases are built gradually over time to make the GP search more amenable, (b) to study for the first time the impact that fitness cases have in DOPs, and to this end, we also use the use of pair-wise fitness disagreements, based on the Kendall Tau distance, as a metric to promote diversity, which has constantly been reported beneficial in DOPs [16].

2 Related Work

2.1 Fitness Cases in Genetic Programming

As discussed previously, the fitness of a GP individual is normally computed by using a set of fitness cases and the way it is used is highly important in GP. The

size of the fitness cases vary e.g., in [15] the authors reported problems that use a small size of fitness cases up to dozens of thousands of fitness cases.

To decide on the number of fitness cases that a GP system may need to use, the rational allocation of trials algorithm can be a good alternative [19]: before a new generation takes place, individuals are evaluated using only a fraction of all the fitness cases available to the system. GP programs are further evaluated on new fitness cases when e.g., there is a possibility of winning some selection mechanism (e.g., tournament selection) that they are losing.

Another approach to determine the necessary number of fitness cases to solve a given problem in GP is that based on well-known statistical and information-theoretic considerations e.g., Central Limit Theorem and entropy of random variables [6]. The authors tested their theoretical framework on discrete fitness-valued cases and showed that their estimations agree with experimental results. Specifically, they showed that when the GP system uses at least the estimated number of fitness cases yield by their approach, the system achieves reliable results and the opposite is true when a lower number of fitness cases is used.

When having a large number of fitness cases, it may be necessary to adopt a mechanism to determine how many and which cases to use. Multiple works have been proposed. For instance, a topology-based mechanism [11] promotes the use of certain fitness cases based on how well or bad these are solved by individuals; historical subset selection [5] uses part of the set of all fitness cases based on how well the elitist individual is able to solve them; active data selection [22] uses small training case sizes and during search, these subsets are recombined and enlarged by a few fitness cases taken from the entire set.

Other sampling methods have been proposed that use a single fitness case in some generations and the entire training set is used in others. Such is the case for Interleaved Sampling and Random Interleaved Sampling [7]. More recently, the Lexicase selection algorithm has been proposed [18], wherein fitness cases are randomly shuffled at each parent selection event, and the best performing individual on the first fitness cases is kept. The method was extended to real-valued problems [10], with performance improving in almost all cases. An evaluation of some of these techniques, as well as others, is reported in [14].

In this work, we take a different approach: rather than determining how many FC the GP system should use, we use them all. The rationale for doing so it is because often there are a few fitness cases available to the system. The novelty of our approach is that instead of presenting all the cases to the system as traditionally done in GP, we build by aggregation these fitness cases over time with the hope to make the search more amenable. Moreover, as indicated before, there are no studies on the impact of fitness cases in DOPs and this works also considers this scenario. It is well-known that diversity plays an important role in evolutionary search, in general, and in DOPs in particular [16]. We present some works on this area next.

2.2 Promoting and Maintaining Diversity

Multiple works have been proposed to promote and maintain diversity in EAs. In this section, we focus only on DOPs tackled by GP (see [16] for a more

general discussion on the subject). Among those approaches proposed to promote diversity in the face of DOPs using GP are: (a) adaptable genetic operators, (b) behavioural diversity, and (c) injection of new genetic structural material. In this work, we only focus on the latter and briefly discuss some approaches that have been proposed to promote diversity via the injection of new individuals.

One of the easiest forms of promoting diversity is adopting the injection of new genetic material into the GP population. The generation of GP individuals is done by using common techniques, like the adoption of the ramped half-and-half method [9]. This can take place when, for instance, detecting a change [13] or when bloat (dramatic increase of tree sizes as evolution proceeds) reaches a limit and there is a need to substitute individuals contained in the population by new GP programs [21]. Injecting new GP individuals into the population has also been promoted via culling [17]. That is, removing the worst individuals and replacing them by randomly generated programs. Variable population size [20] also promotes diversity by adding new GP individuals into the population.

A common element in all these works that promote structural diversity is that the number of individuals to be replaced by the same number of newly created individuals is chosen rather arbitrarily. Next, we present an approach that aims to overcome this limitation.

3 Proposed Approaches

As discussed previously, we are interested in making the GP search more amenable and to do so we propose a dynamic fitness cases approach, wherein cases are built by aggregation over time. We test this approach in both static and DOPs, and for the latter, we also use the adoption of the Kendall Tau Distance (KTD) that quantifies pairwise dissimilarities among two lists of fitness values with the hope to make a better informed decision in terms of the number of individuals that need to be replaced in a population by new individuals to promote diversity.

3.1 Dynamic Fitness Cases

To make the GP search more amenable, we build the fitness cases over time. More specifically, at the beginning of an evolutionary run or just after a change has occurred (for the dynamic setting), we use in order a subset of fitness cases, $C_{g=0}$ which is chosen from all the fitness cases C^N of size N,

$$C_{g=0} \subset C^N, |C_{g=0}| = k \tag{1}$$

where k is a constant and $k < N$. After a few i generations another k fitness cases of the C^N fitness cases are added to $C_{g=0}$,

$$C_{g=0} \cup C_{g=i}, C_{g=0} \cap C_{g=i} = \{\} \tag{2}$$

We continue this process until all the fitness cases have been used. Thus, the complete sequence of fitness cases is build as follows,

$$C_{g=0} \cup C_{g=i} \cup \cdots \cup C_{g=M} = C^N \tag{3}$$

where M is a constant and $M < K$, where K is either the maximum number of generations or the number of generations that are necessary for a change to take place (for the dynamic scenario). By defining the latter, we guarantee that the GP system accounts for all the fitness cases before a change takes place and it has all the necessary elements to, potentially, find the solution. The values of the variables are defined in Table 2 and discussed in Section 5.

3.2 Kendall Tau Distance

As indicated before, there is no study that has focused its attention on the study of fitness cases in a dynamic setting and this work also considers such scenario.

As seen in Section 2, we know that there is strong evidence indicating that the adoption and/or encouragement of diversity in GP search on DOPs is highly beneficial. Normally, when adopting this type of diversity, researchers have focused their attention on setting arbitrarily a number of individuals to be generated and then used them to e.g., replace the worst GP individuals in a population. The major drawback with this approach is that often this process is based on trial and error and can be computationally expensive.

We believe that it is possible to adopt a more informed way of determining the number of individuals that should be replaced from a population by using fitness values. The use of these values as indicators to perform a specific task (e.g., prediction of problem hardness) is common in EAs. The most well-known example of this is the fitness-distance correlation [8], where these values are used in conjunction with a metric that informs us how distant two individuals are in the search space to determine problem difficulty.

In this work, we use a distance, studied in the first author's works [1, 3], that accounts for pairwise disagreements between two lists of ranked fitness values. We hope that these disagreements can inform us on whether an evolved population is useful in the face of a change. Our proposed approach works in three phases:

1. Firstly, it is necessary to account for a method that can indicate when a change is about to take place. We do this in a non-expensive manner: before a new generation is about to take place, we use one individual (the elitist individual), whose fitness (f_e^g) is assessed again in the next generation (g+1).
2. Secondly, if f_e^g and f_e^{g+1} are different, then we regard this as a change in the environment and we then proceed to compute the KTD (defined in Eq. 4) between the ranking of the fitness values of all individuals at generation g and the next generation ($g+1$). This distance counts the number of pairwise disagreements between two ranked lists and it is normalised by the maximum number of possible disagreements. This distance gives a discrete value $k = [0, 1]$ and this is used to generate a percentage of T new individuals with respect to the population size.
3. Thirdly, the worst (less fit) individuals at g+1 are replaced by the newly generated individuals (using ramped half-and-half initialisation method, details are discussed in Section 4) keeping the size of the population constant.

The KTD between two ranked lists is defined as,

Table 1. Symbolic regression benchmarks problems used in our work.

Function	Objective function
f_1	$x^3 + x^2 + \alpha x$
f_2	$x^4 + x^3 + x^2 + \alpha x$
f_3	$x^5 + x^4 + x^3 + x^2 + \alpha x$
f_4	$x^6 + x^5 + x^4 + x^3 + x^2 + \alpha x$
f_5	$\sin(x^2)\cos(\alpha) - 1$
f_6	$\sin(\alpha x) + \sin(x + x^2)$
f_7	$\log(\alpha x + 1) + \log(x^2 + 1)$
f_8	$\text{sqrt}(\alpha x)$

Table 2. Summary of Parameters.

Parameter	*Value*
Population Size	800
Generations	200
Type of Crossover	Any node
Crossover Rate	0.80
Type of Mutation	Subtree
Mutation Rate	0.20
Selection	Tournament (size = 7)
Initialisation Method	Ramped half-and-half
Initialisation Depths:	
Initial Depth	2
Final Depth	5
Maximum Length	1200 nodes
Maximum Final Depth	8
Independent Runs	50
Changes	Every 50 generations
Dynamic Fitness Cases	$k = 1$, $i = 2$, $M = 39$

$$k(\tau_1, \tau_2) = \sum_{(i,j) \in P} \bar{k}_{i,j}(\tau_1, \tau_2) \tag{4}$$

where, P is the set of pairs of elements in τ_1 and τ_2, $\bar{k}_{i,j}(\tau_1, \tau_2) = 0$ if i and j are in the same order in both τ_1 and τ_2; and 1 if i and j are in opposite order.

It is worth mentioning that when the change to the objective function is monotonically increasing (order preserving), the computed KTD will be 0. This is a good property because in this case the evolved individuals are expected to behave well in the changed objective function, so there is no need to replace individuals. A mirror image is seen in the presence of a monotonically decreasing change of the objective function, which will yield the maximal normalised distance of 1, meaning that the order of both fitness lists is completely different. The latter will indicate that our approach based on the KTD will replace the entire population by newly generated genetic material.

4 Experimental Setup

To test our approach, we use eight symbolic regression functions of various difficulties, shown in Table 1. The fitness function is computed as one over one plus the sum of absolute errors of the Euclidean distance to the output vector of the target uni-variate function queried on 20 inputs in the equally drawn range [1, 1]. Our system maximises it. A solution is regarded as correct when its fitness is greater or equal than 1 - 0.01. The function set is F = {+,-,*,/}, where / is protected division.

To test separately and in conjunction our two approaches, we use a static and a dynamic setting. We define three different type of changes for the latter: we use α as a variable (see Table 1) that can be tuned to achieve this along with a constant L, set at 50, that denotes when α changes to simulate a change (in this work, the maximum number of generations is set at 200, hence only three

Table 3. Success rate (%) and avg. of best fitness using either static (SFC) or dynamic fitness cases (DFC). No changes take place during evolution. All the results on the avg. of best fitness are statistical significant (Wilcoxon Test at 95% level of significance). Higher is better.

Function	Success Rate		Avg. Best Fitness	
	SFC	DFC	SFC	DFC
f_1	92.0%	100.0%	0.9371	1.0000
f_2	54.0%	88.0%	0.6656	0.9969
f_3	18.0%	70.0%	0.4501	0.9915
f_4	4.0%	72.0%	0.3280	0.9895
f_5	0.0%	60.0%	0.4580	0.9896
f_6	0.0%	64.0%	0.3438	0.9893
f_7	0.0%	36.0%	0.4988	0.9739
f_8	0.0%	16.0%	0.3068	0.9665

values for α are required for a dynamic setting, as defined next). For the static scenario, $\alpha = 1$. For the dynamic setting, we define a smooth, an 'abrupt' and a random change, where $\alpha = \{0.9, 0.8, 0.7\}$, $\alpha = \{0.1, 0.9, 0.1\}$, and finally, α is set with a random value between 0 and 1 every L generations, respectively.

For comparative purposes, we use a static fitness case-scenario and our proposed dynamic fitness case-approach, where all the cases are presented to the system at the beginning of the search as commonly adopted in the GP community and where the cases are built over time, respectively.

Moreover, for the DOPs defined in this work, we use an arbitrary approach, wherein the number of individuals to be replaced in a population is generated randomly and compared it against the results yield by our proposed Kendall Tau distance. We generate the individuals in these two approaches using the ramped half-and-half method, where the initial and final depths used are the same as when generating the population (see Table 2).

The experiments were conducted using a generational approach. The parameters used are shown in Table 2. To obtain meaningful results, we performed an extensive empirical experimentation (50 * 2 * 3 * 8 runs, plus 50 runs for each fitness case scenario: static and dynamic; 2,500 independent runs in total)[5].

5 Results and Discussion

5.1 Performance on a Static Setting

Let us analyse the results when using our proposed dynamic fitness cases (DFC), wherein cases are built over time, where one fitness case is added every two generations until all of them have been used, as defined at the bottom of Table 2, (see Section 3 for details on how this works) and compared the results obtained by DFC against the widely adopted mechanism of using all the fitness cases at the beginning of the search, denominated in this work as static fitness cases (SFC).

[5] 50 independent runs, 2 types of replacement of individuals (arbitrary, Kendall tau distance-based), 3 types of changes, 8 problems.

Table 3 shows the success rate (shown in the 2nd and 3rd column, from left to right), defined as the number of times that the GP system was able to find the solution and the average of the best fitness at the end of each independent run (shown in the last two columns).

It is clear to see that the proposed DFC achieves good results in terms of finding the solution, as indicated in Table 3. The traditional SFC has a good performance only on the relatively easy f_1 and its performance decreases significantly with the rest of the functions used in this work, where SFC is not able to find a single solution for functions f_5, f_6, f_7 and f_8 in any of the independent runs. Our proposed DFC, on the other hand, achieves better results e.g., 60%, 64%, 36% and 16% for functions f_5, f_6, f_7 and f_8, respectively.

The results shown in the last two columns of Table 3, which are the average of the best individuals' fitness values at the end of each run, are aligned to the performance achieved by SFC and FDC. These results are all statistically significant, Wilcoxon Test set at 95% level of significance.

5.2 Performance on a Dynamic Setting

Let us first focus our attention on the performance achieved by the static and the dynamic function case-based approach, when these two are now used in conjunction with our proposed Kendall Tau Distance (KTD) approach to promote diversity in three type of changes: smooth, random and 'abrupt' change, as defined in Section 4.

These results, shown in Table 4, are similar to those discussed above: the SFC approach has a poor performance: less than 3.0%, for functions f_5 - f_8, defined in Table 1, regardless of the type of change used. These results are significantly better when using the proposed DFC in conjunction with the KTD approach. For example, the proposed approach achieves more than 48%, 67%, 59% and 52% for the same referred functions, respectively, regardless of the change used.

The average of best fitness values just before a change takes place, defined at every 50 generations, is shown in Table 5. These results (all statistically significant, Wilcoxon Test set at 95% level of significance) are aligned to the performance discussed above: the average fitness values is poor when using SFC compared to those results achieved by DFC. For example, for f_3 the average fitness values achieved by our proposed DFC is around 0.93 (almost three times better compared to the results yield by SFC), regardless of the change used.

Now, let us discuss the results when using the commonly adopted approach of replacing a random number of individuals from a population to promote diversity, referred in this work as the arbitrary approach. These results are shown in Tables 6 and 7 for the percentage of success rate and the average of the best fitness values just before a change occurs, respectively. In these tables, we can observe a similar scenario compared to what we discussed when using the KTD for an informed way to replace a number of individuals. That is, the SFC approach yields significantly worse results compared to DFC.

If we now compare, for instance, the performance achieved by the KTD and the arbitrary approach focusing on either using static fitness cases or using

Table 4. Percentage of success rate using both static and dynamic fitness cases on eight different regression functions in the presence of three different types of changes. Replacement type used: Kendall approach.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	21.5%	25.0%	24.5%	33.5%	21.5%	29.0%
f_2	10.0%	92.5%	11.0%	90.0%	10.5%	91.5%
f_3	2.5%	79.0%	4.5%	89.0%	2.5%	82.0%
f_4	0.5%	87.0%	1.5%	86.0%	0.5%	90.5%
f_5	0.0%	49.0%	0.0%	60.5%	0.0%	57.0%
f_6	0.0%	68.0%	0.5%	.80.5%	0.0%	77.5%
f_7	0.0%	76.0%	0.0%	.80.0%	2.5%	60.0%
f_8	0.0%	53.0%	0.5%	64.0%	1.0%	61.0%

Table 5. Avg. of best fitness values at every 50th generation (just before a change takes place) using both static and dynamic fitness cases on eight symbolic regression functions in the presence of three different types of changes. Replacement type used: Kendall approach. All the results are statistical significant (Wilcoxon Test at 95% level of significance). Higher is better.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	0.482	0.7594	0.5444	0.8115	0.5574	0.7583
f_2	0.4245	0.9572	0.4913	0.9566	0.5088	0.9648
f_3	0.3372	0.9457	0.3886	0.9547	0.4420	0.9508
f_4	0.3332	0.9435	0.3862	0.9512	0.4218	0.9589
f_5	0.3326	0.8901	0.4153	0.9129	0.4122	0.9156
f_6	0.3099	0.9218	0.3939	0.9397	0.4470	0.9401
f_7	0.4639	0.9249	0.5006	0.9312	0.5092	0.9042
f_8	0.3288	0.8831	0.4113	0.9010	0.4453	0.8968

Table 6. Percentage of success rate using both static and dynamic fitness cases on eight different regression functions in the presence of three different types of changes. Replacement type used: Arbitrary approach.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	21.5%	25.0%	24.5%	35.0%	21.5%	33.5%
f_2	10.0%	90.0%	11.0%	92.5%	10.5%	95.0%
f_3	2.5%	84.0%	3.5%	85.0%	2.5%	85.0%
f_4	0.5%	89.5%	1.5%	87.5%	0.5%	90.0%
f_5	0.0%	47.0%	0.5%	61.0%	0.5%	56.0%
f_6	0.0%	73.5%	0.5%	80.5%	0.5%	85.5%
f_7	0.0%	75.0%	0.5%	74.5%	6.5%	71.0%
f_8	0.0%	55.5%	0.0%	67.5%	1.0%	66.5%

Table 7. Avg. of best fitness values at every 50th generation (just before a change takes place) using both static and dynamic fitness cases on eight symbolic regression functions in the presence of three different types of changes. Replacement type used: Arbitrary approach. All the results are statistical significant (Wilcoxon Test at 95% level of significance). Higher is better.

Function	Smooth Change		Random Change		Abrupt Change	
	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases	Static Cases	Dynamic Cases
f_1	0.4793	0.7714	0.567	0.8099	0.5889	0.8337
f_2	0.4288	0.9561	0.5089	0.9638	0.5160	0.9701
f_3	0.3337	0.9477	0.4059	0.9521	0.4705	0.9563
f_4	0.3439	0.9514	0.3887	0.9538	0.4394	0.9628
f_5	0.3444	0.8927	0.4228	0.9098	0.4257	0.9202
f_6	0.3478	0.9342	0.4108	0.9424	0.4747	0.9514
f_7	0.4675	0.9285	0.5137	0.9296	0.5397	0.9214
f_8	0.3282	0.8863	0.4149	0.9121	0.4622	0.9072

dynamic fitness cases, we do not see much difference. For example, the performance for the function f_5 is 49.0% (Table 4) and 47.0% (Table 6), using the KTD approach and the arbitrary approach in the presence of a smooth change, respectively. The same trend is observed for the rest of the functions regardless of the type of change used. However, the benefit of using the KTD in DOPs instead of using an arbitrary approach (random number of individuals replaced in a population), as normally adopted in EAs DOPs when promoting diversity via the replacement of individuals, can be observed when analysing the number of individuals created by either approach. We discuss this next.

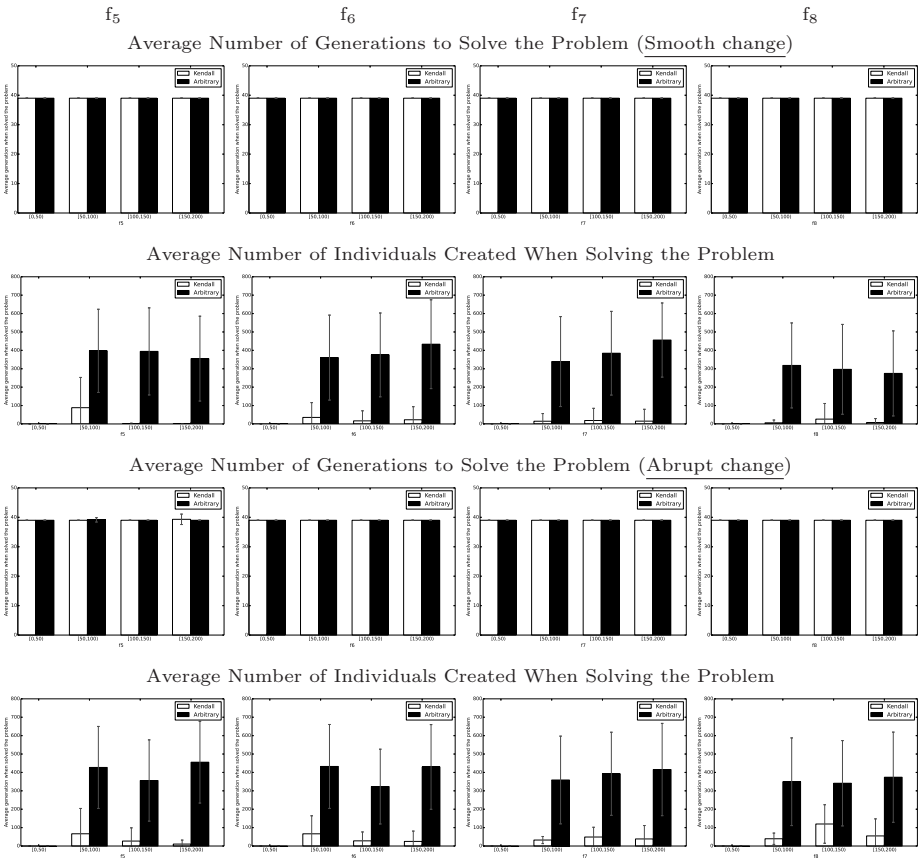

Fig. 1. Average number of generations to solve a problem (odd rows) and average number of created individuals (even rows) along with standard deviation using either the arbitrary approach (black-filled rectangle) or our proposed Kendall approach (white-filled rectangle) for functions f_5 - f_8 when using dynamic fitness cases. Notice that for the avg. number of individuals created, the first generations [0,50), show nothing given that a change occurs after this.

5.3 Analysis of the Number of Created Individuals

To see the benefit of using the proposed KTD approach in DOPs compared to the arbitrary approach to promote diversity via the replacement of individuals in the population by new genetic material, it is necessary to see the number of individuals created by each of these two approaches. This is shown in the second and fourth rows, from top to bottom, of Figure 1, for functions f_5 - f_8, where the vertical line denotes the standard deviation. Due to space constraints, we only show the results when using the DFC approach on these functions and in the presence of a smooth and an 'abrupt' change that yield better results compared to the SFC approach. However, a similar trend was observed for the rest of the functions and type of change.

Let us discuss a particular example: when a smooth change takes place for functions f_5 - f_8, shown in the second row of Figure 1. It is clear to see that the number of individuals created by the KTD, shown by a white-filled bar is significantly lower compared the number of GP programs created by the arbitrary approach, shown by a black-filled bar. This is to be expected since a smooth change took place and our proposed approach was able to capture this by creating a few individuals in the presence of this type of change. The same trend is observed for the rest of the functions (not shown due to space constraints).

Moreover, we can see that the KTD approach is able to capture the level of a change. For instance, see the number of created individuals in the presence of an smooth change *vs.* an 'abrupt' change, as denoted in the second and fourth row of Figure 1: the KTD approach creates less number of individuals in the presence of a smooth change compared to the 'abrupt' change. Although the difference of created individuals in the presence of either these two changes is small. This is due to the type of changes proposed in this work (see Section 4) rather than the KTD approach failing at capturing a change. To illustrate this, we adopted a more radical change where the last sign in f_1 changes every 50 generations (from '+' to '-' and *vice versa*) to simulate a DOP and compare this result against those yield by the other three types of changes. This is shown in Figure 2, where we can clearly see that the KTD yields values accordingly: the number of created individuals is increased as the change is more severe.

In addition to this, we also analyse the average number of generations required to solve a problem. This is shown in the first and third row in Figure 1. Interestingly, we can see that the majority of problems, regardless of the change used, are solved once all the fitness cases are presented to the system: observe how they finish before generation 40 (recall that all fitness cases are presented to the GP system at generation M =39 as indicated in Table 2), with a few runs finding the solution just after generation 40 as denoted by the small standard deviation (see, for example f_5, in the presence of an abrupt change, third row first column of Figure 1).

5.4 Size of GP Programs

We have learnt that DFC behaves better than the widely-adopted SFC in GP. We believe that the reason is due to the fact that GP system gradually solves

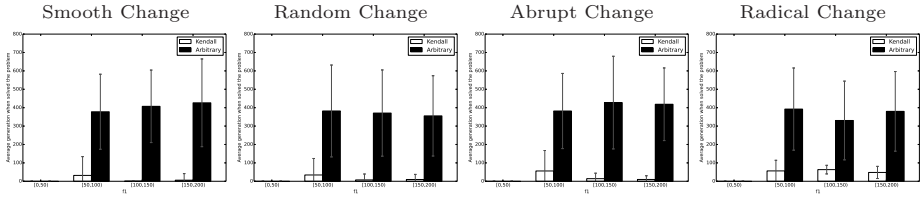

Fig. 2. Average number of created individuals along with standard deviation using either the arbitrary approach (black-filled rectangle) or our proposed Kendall approach (white-filled rectangle) for function f_1 when using static fitness cases. Notice that for the average of created individuals, the first generations [0,50) show nothing, given that a change occurs after this.

the problem in accordance to the proposed DFC, wherein fitness cases are built by aggregation over time (see Section 3). This in consequence could mean that the GP program gradually starts growing as more cases are presented to the system. Indeed, this is what can be observed in Fig. 3, where we report the average length of individuals, along with the standard deviation, on f_3 and f_4 using the arbitrary replacement approach and an abrupt change (the same trend is observed for the other two type of changes and replacement mechanisms not shown due to space constraints). It is evident that the size of programs created by the DFC approach, denoted by grey lines, is significantly lower compared to the traditional SFC approach, indicated by black lines.

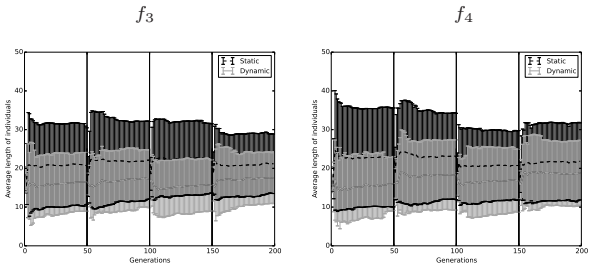

Fig. 3. Average (along with a standard deviation) length of programs using both static and dynamic fitness cases, shown in black and grey lines, respectively, on f_3 and f_4, using the arbitrary replacement approach. Vertical lines at every 50th generations indicate an (abrupt) change.

6 Conclusions

Traditionally, the fitness value of a GP program is computed by using a set of fitness cases. It is common that all the fitness cases are presented to GP from the beginning of the search, an approach we call static fitness cases. In this

work, we propose a dynamic fitness cases approach, wherein the cases are built by aggregation over time, making it an incremental search. We showed that the proposed approach achieves better better performance, in some problems achieving a 60% success rate compared to 0% achieved by the standard approach.

Furthermore, we tested these two approaches in the presence of dynamic changes, where the results achieved by the DFC are consistently better compared to the SFC. Moreover, we also showed how the DFC approach encourages a smooth increase of GP trees compared to SFC where the size of trees are bigger.

Finally, we also studied the impact/use of fitness cases in DOPs, where the adoption of diversity has consistently been reported as beneficial. To this end, we proposed an approach based on the Kendall Tau Distance that aims to capture the degree of a change in a dynamic setting and we use this consequently to determine the proportion of individuals that need to be replaced to promote structural diversity. We compared this against the commonly adopted arbitrary approach where the number of individuals is set randomly. We showed that the performance of both replacement mechanisms is similar, with the added benefit that the proposed KTD approach creates only the necessary individuals with regards to the amount of change.

Acknowledgments. EGL would like to thank the TAU group at INRIA Saclay for hosting him during the outgoing phase of his Marie Curie fellowship and for financially supporting him to present this work at the conference. LT would like to thank CONACYT (project FC-2015-2:944) for providing partial funding.

References

1. E. Galván-López and O. Ait ElHara. Using fitness comparison disagreements as a metric for promoting diversity in dynamic optimisation problems. In *IEEE Symposium Series on Computational Intelligence*. Springer, 2016.
2. E. Galván-López, E. Mezura-Montes, O. Ait ElHara, and M. Schoenauer. On the use of semantics in multi-objective genetic programming. In J. Handl et al., editors, *Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*, pages 353–363. Springer, 2016.
3. E. Galván-López, L. Vázquez-Mendoza, M. Schoenauer, and L. Trujillo. Dynamic GP fitness cases in static and dynamic optimisation problems. In P. A. N. Bosman, editor, *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 227–228. ACM, 2017.
4. E. Galván-López, L. Vázquez-Mendoza, and L. Trujillo. Stochastic semantic-based multi-objective genetic programming optimisation for classification of imbalanced data. In O. Pichardo-Lagunas and S. Miranda-Jiménez, editors, *Advances in Soft Computing*, chapter 22, pages 261–272. Springer, 2016.
5. C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In Y. Davidor et al., editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 312–321, Jerusalem, 9-14 Oct. 1994. Springer-Verlag.
6. M. Giacobini, M. Tomassini, and L. Vanneschi. Limiting the number of fitness cases in genetic programming using statistics. In *Parallel Problem Solving from Nature VII*, pages 371–380, London, UK, 2002. Springer-Verlag.

7. I. Gonçalves and S. Silva. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In K. Krawiec et al., editors, *Genetic Programming*, volume 7831 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg, 2013.
8. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.
9. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
10. W. La Cava, L. Spector, and K. Danai. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 741–748, New York, NY, USA, 2016. ACM.
11. C. W. G. Lasarczyk, P. W. G. Dittrich, and W. W. G. Banzhaf. Dynamic subset selection based on a fitness case topology. *Evol. Comput.*, 12(2):223–242, June 2004.
12. U. López, L. Trujillo, Y. Martinez, P. Legrand, E. Naredo, and S. Silva. Ransacgp: Dealing with outliers in symbolic regression with genetic programming. In J. McDermott et al., editors, *Genetic Programming: 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*, pages 114–130, Cham, 2017. Springer International Publishing.
13. J. Macedo, E. Costa, and L. Marques. *Genetic Programming Algorithms for Dynamic Environments*, pages 280–295. Springer, Cham, 2016.
14. Y. Martínez, E. Naredo, L. Trujillo, P. Legrand, and U. López. A comparison of fitness-case sampling methods for genetic programming. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 1–22, 2017.
15. J. McDermott et al. Genetic programming needs better benchmarks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 791–798, New York, NY, USA, 2012. ACM.
16. T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm Evol. Comput.*, 6:1 – 24, 2012.
17. M. Riekert, K. M. Malan, and A. P. Engelbrect. Adaptive genetic programming for dynamic classification problems. In *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*, CEC'09, pages 674–681, Piscataway, NJ, USA, 2009. IEEE Press.
18. L. Spector. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, GECCO Companion '12, pages 401–408. ACM, 2012.
19. A. Teller and D. Andre. Automatically choosing the number of fitness cases: The rational allocation of trials. In J. R. Koza et al., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 321–328, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.
20. L. Vanneschi and G. Cuccu. A study of genetic programming variable population size for dynamic optimization problems. In *IJCCI*, pages 119–126, 2009.
21. N. Wagner, Z. Michalewicz, M. Khouja, and R. R. McGregor. Time series forecasting for dynamic environments: The dyfor genetic program model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452, Aug 2007.
22. B.-T. Zhang and D.-Y. Cho. Genetic programming with active data selection. In *Simulated Evolution and Learning on Simulated Evolution and Learning*, SEAL'98, pages 146–153, London, UK, 1999. Springer-Verlag.