Benchmarking Pareto Archiving heuristics in the presence of concept drift: Diversity versus Age

Aaron Atwater[1] and Malcolm I. Heywood[1]

[1]*Faculty of Computer Science, Dalhousie University, Halifax, NS. Canada*

Article originally appears at GECCO under ACM copyright 2013
http://dl.acm.org/citation.cfm?doid=2463372.2463489

Abstract

A framework for coevolving genetic programming teams with Pareto archiving is benchmarked under two representative tasks for non-stationary streaming environments. The specific interest lies in determining the relative contribution of diversity and aging heuristics to the maintenance of the Pareto archive. Pareto archiving, in turn, is responsible for targeting data (and therefore champion individuals) as appropriate for retention beyond the limiting scope of the sliding window interface to the data stream. Fitness sharing alone is considered most effective under a non-stationary stream characterized by continuous (incremental) changes. Fitness sharing with an aging heuristic acts as the preferred heuristic when the stream is characterized by non-stationary stepwise changes.

1 Introduction

Streaming data applications represent a requirement for online as opposed to offline evolution. Taken from the perspective of a classification or regression task this implies that it is not possible to assume that the data can be divided into independent training, validation and test partitions i.e., the underlying process (describing the data) is non-stationary. This means that evolution should be conducted as a continuous process. The principle interest is therefore with respect to how well a machine learning algorithm – genetic programming (GP) in this case – is capable of tracking / reacting to such changes.

Streaming applications appear frequently in financial environments (e.g., [7]), computer network applications (e.g., [27]) or 'big data' applications (e.g., [2]). The interface between data stream and evolutionary algorithm typically takes the form of a sliding window. At each training epoch (generation) only data within the window can be accessed and used for fitness evaluation. In this work, evolution under a Pareto archiving model is assumed (Section 3). The implication of this is that a formal framework exists for identifying and retaining the (hopefully) few instances from the stream that are most useful in promoting the identification of the fittest learners.

The explicit interest of this research is to characterize the suitability of heuristics used to maintain finite archiving constraints necessary to minimize computational overheads associated with Pareto archiving. Such heuristics have a direct impact on the capability of the system to operate within non-stationary environments. In particular, we are interested in quantifying the importance attributed to heuristics for diversity versus age. With this in mind, two artificial data generators will be employed to create multi-class classification benchmarks with stepped versus continuous variation of the underlying data generating process (Section 4). Section 2 will summarize related research in dynamic environments whereas Section 3 summarizes the approach to Pareto archiving and team based task decomposition in GP. The paper concludes with a discussion and an identification of future research in Section 5.

2 Related work

Unlike static tasks, dynamic or non-stationary tasks require convergence to be pursued only up to a point. Specifically, when the underlying task is dynamic, convergence potentially compromises the ability to react to change. It has been proposed that previous approaches investigated for applying evolutionary computation to dynamic environments fall into one of five forms [7]: memory, diversity, multi-populations, problem decomposition and

evolvability. The following provides a short review of the contributions made and where necessary their relation to this work.

Memory: as reflected in the capability to return to a previously evolved solution. This implicitly assumes that the task is in some way periodic. Moreover, there is always a tradeoff in terms of the amount of resources made available for memory and the amount made available for supporting other properties, such as diversity. In this work we are interested in the utility of Pareto archiving as a candidate memory mechanism under explicit enforcement of finite archive sizes. Pareto archiving provides a much more formal basis for retaining both learners (GP individuals) and points (data 'distinguishing' between non-dominated learners). A previous article looked at the simplified case of Pareto archiving applied under a streaming data context, but with respect to an environment with a underlying dynamic that was stationary [2]. As such the earlier work was not able to draw any conclusions regarding the appropriateness of Pareto archiving to non-stationary environments.

Diversity: as in mechanisms for supporting multiple species within a population. The most typically assumed mechanism is to vary the rate of mutation or introduce a fixed number of entirely new individuals at each generation [13]. Niching (speciation) has been considered to resist favouring the same individuals [5]. Likewise, solution age has been proposed to bias the retention of the 'middle aged' as opposed to the old or new [12]. [20] introduced 'sentinel' individuals as those which are uniformly distributed through the representation space. Such individuals act as a diverse source of genotypic information for seeding the population on a continuous basis. The later benchmarking study conducted here will revisit and compare the advantages and disadvantages of assuming speciation and age heuristics in conjunction with Pareto archiving.

Multi-populations: associate different populations with different aspects of the search space. The basic assumption in this case is that genetic drift associated with the smaller subpopulations will encourage the investigation of new areas of the search space. Finding a suitable heuristic for controlling the exploration–exploitation tradeoff controlling the interaction between the multiple populations remains an ongoing research issue.

Problem decomposition: is taken to have been articulated first by Simon's watchmaker parable in which the capacity to configure quickly is attributed to having appropriate 'modules' available [26]. From the perspective of GP, task decomposition might be most synonymous with support for modularity, run–time libraries and teaming. Studies have shown the utility of each relative to static task domains—e.g., in the case of the teaming metaphor adopted here, see [18].

Evolvability: implies that a representation is assumed that supports the identification of fitter parents in changing environments. Geno- to phenotypic mappings have been proposed in this respect e.g., [24]. In this work we equate evolvability with the degree to which modularity is supported, and a symbiotic coevolutionary framework for teaming in GP specifically [18].

3 Pareto archiving in GP

An noted in Sections 1 and 2, Pareto archiving will be assumed as the 'memory mechanism' by which the uniqueness of GP individuals (learners) is formally established. Potentially, Pareto archiving provides a scheme for identifying a subset of learners as being non-dominated relative to the rest of the population. To do so, training exemplars represent 'objectives' and are rewarded for distinguishing between the capability of different learners. Such a model is of interest in a streaming context as a basis is then provided for identifying a small number of data instances for retention beyond the immediate content of the sliding window i.e., interface to the stream (Section 3.1). Hereafter, the specific form assumed for GP will be the Symbiotic bid-based (SBB) framework for cooperatively evolving GP teams [17, 8]. Such a framework supports task decomposition and multi-class classification from a single evolutionary cycle and is therefore potentially capable of supporting multiple factors significant to dynamic environments (as reviewed in Section 2). In the following we first develop the Pareto archiving methodology independent of GP (Section 3.2) and then summarize SBB (Section 3.3).

3.1 Streaming data and concept drift

The streaming data assumption used in this work is the same sliding window implementation used in previous work such as [2]. This implies that at any given generation t only a subset of the total training instances are available to be sampled from, and at generation $t+1$ some number of instances will have left the window and will never be accessible again, while simultaneously some previously inaccessible instances will also be added to the window and thus available for selection. As in previous cases, the GP system is provided time equivalent to 10% of the maximum number of generations, t_{max}, with access to the first 10% of the stream, in order to train an initial population of GP individuals.

Concept drift is introduced in the data sets used herein, in the incarnation of multiple distinct rulesets which partition the attribute space into separate classes being employed at different time points throughout the stream. Two distinct rulesets may or may not be conceptually related, and data representing both these cases is presented in Section 4.2. The challenge posed to an online learning system in such an environment is to not only correctly determine the correct class label for exemplars in the current environment, but also to quickly adapt to changes in the underlying concepts – something that cannot be explicitly trained upon in the sense of traditional supervised learning tasks.

3.2 Pareto archiving and fitness sharing

A two stage process is assumed consisting of Pareto dominance ranking and fitness sharing,[1] the latter representing a heuristic for promoting diversity when enforcing finite archiving constraints.

Pareto dominance ranking: The members of the point population, p_k are taken to represent 'objectives'. Fitness is only ever evaluated against the content of the point population. This decouples fitness evaluation from the cardinality of a data set. From a streaming data perspective P_{gap} points are replaced at each generation with a sample of new points taken from the current sliding window location [2]. Such objectives / points are used to distinguish between the capability of different GP learners under a pairwise test for Pareto dominance, or

$$\begin{array}{c} \forall p_k \in P : G(l_i, p_k) \geq G(l_j, p_k) \\ \text{AND } \exists p_k \in P : G(l_i, p_k) > G(l_j, p_k) \end{array} \tag{1}$$

where P is the point population; l_i and l_j are two individuals (cf., learners) from the GP population currently under evaluation, and; $G(\cdot, \cdot)$ is the task specific reward function returning 0 on an incorrect classification and 1 on a correct classification (see Eqn (4) later).

Eqn (1) implies that learner l_i dominates learner l_j *iff* it performs as well on every point and better on at least one point. For a total of $|L|$ learners there are a total of $|L|^2 - |L|$ learner comparisons [6].[2] Moreover, for point p_k a comparison between learner l_i and l_j has the form of a distinction vector:

$$d_k[L \cdot i + j] = \begin{cases} 1 & if G(l_i, p_k) > G(l_j, p_k) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Thus, when Eqn (2) returns a value of 1 then a *distinction* is said to have been made [10]. Points are penalized for defeating all learners or when it is defeated by all learners i.e., no distinctions result. Such points would therefore tend to be prioritized for removing from the point population. The fittest learners / GP individuals are naturally those promoted by the Pareto dominance expression – Eqn (1). One drawback of the Pareto approach is that as the number of objectives (points) increases, then comparatively weak overlapping behaviours may satisfy the Pareto archiving criteria [10, 21, 19]. SBB therefore adopts the following fitness sharing heuristic with the motivation of maintaining diversity in the points learners solve. This point will be revisited in Section 4 when we consider alternative heuristics that might be more appropriate to a data streaming context.

Fitness sharing: The reward function $G(l_i, p_k)$ establishes a vector of distinctions (Eqn. (2)) for each point. The point population can now be divided into two sets: those in the non-dominated front (the archive), $\mathcal{F}(P^t)$, versus those that are not, $\mathcal{D}(P^t)$. Naturally, the decision to limit the number of archives to two is motivated by a desire to balance the computational cost of archive construction against maintaining monotonic progress to an 'ideal' training trajectory [6].

Up to P_{gap} points are replaced at each generation from the current stream window using the process of Section 3.1 [2]. Points are targeted for replacement under two basic conditions [17, 8]:

- If $|\mathcal{F}(P^t)| \leq |P| - P_{gap}$ THEN: stochastically select points for replacement from $\mathcal{D}(P^t)$ alone.
- If $|\mathcal{F}(P^t)| > |P| - P_{gap}$ THEN: all the dominated points are replaced, plus some subset $P_{gap} - |\mathcal{D}(P^t)|$ of the Pareto archive. A fitness sharing heuristic is assumed for weighting members of the Pareto archive such that the more unique the distinction the greater the weight. Thus, relative to distinction vector, d_k, of point p_k fitness sharing is defined as:

$$\sum_i \frac{d_k[i]}{1 + N_i} \tag{3}$$

[1] For a tutorial on Pareto archiving as applied to GP see [15].

[2] This is distinct from the number of fitness evaluations per generation, with fitness evaluation being a more costly process than establishing the comparison vector.

where i indexes all distinction vector entries (Eqn (2)), and N_i counts the number of points in $\mathcal{F}(P^t)$ that make the same distinction.

3.3 Symbiotic bid-based GP

Symbiotic bid-based GP as applied to classification – hereafter SBB – utilizes two populations: program and team (or symbiont and host); which are coevolved through symbiosis [17, 8]. The program (symbiont) population utilizes a bid-based GP representation [16], itself defining each individual in terms of a tuple $\langle c, b \rangle$. Here c declares a scalar class label selected from the set of labels associated with the task domain ($c \in C$); whereas b is the program that evolves a context for deploying its class label. The team (host) population identifies combinations of symbionts that attempt to coexist in a coevolutionary relationship. Members of the team population assume a variable length representation, wherein the number of symbionts per team are adapted as part of the evolutionary cycle. Indeed, [2] noted that hosts were first composed from symbionts representing the most frequent classes and only later were symbionts added representing the less frequent classes. Fitness evaluation is only ever performed at the 'level' of the hosts; thus, hosts represent the 'learner' in the above discussion of Pareto dominance (Section 3.2).

Evaluation of a host, $l_i \in L^t$, is repeated for each training instance as identified by the point population at generation t, $p_k \in P^t$. For all symbionts a member of the host, $s_j \in l_i$, execute their programs w.r.t. point p_k. The host identifies the symbiont with maximum output (the winning bid) or $s^* = arg_{s_j \in l_i} max[s_j.b]$. The winning symbiont gains the right to suggest the class label for the current point, or

$$G(l_i, p_k) = \begin{cases} 1 & if \text{ IF } s^*.c = p_k.t \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $p_k.t$ is the target label for point p_k. Naturally, $G(l_i, p_k)$ is the reward function referred to in Section 3.2.

Algorithm 1 summarizes the breeder model of evolution defining the stepwise application of SBB to a data stream [2]. Initialization provides initial sliding window content, S^t from the data stream, τ, and then samples from S^t until $P - P_{gap}$ points have been selected (Section 3.1). Initialization of the host and symbiont population follows the original SBB algorithm [17, 8]. On entering the main loop the remaining P_{gap} and L_{gap} points and teams are initialized. The point population assumes the same stochastic sampled process as used during the initialization step. Additional L_{gap} hosts are added through application of the SBB variation operators. Variation operators include crossover between hosts and symbiont mutation. The latter implies symbionts are first cloned, with the content varied through appropriate GP mutation operators [17, 8].

The 'Evaluate' function (line 11) resolves values for $G(l_i, p_k)$. Once all hosts have been evaluated on all points then the location of the sliding window is incremented relative to the data stream S^t (line 14). Point replacement is performed under the Pareto archiving–fitness sharing heuristic of Section 3.2 (line 15). Likewise, a fixed number of hosts, L_{gap}, are removed at each generation (line 16) resulting in members of the host population being either non-dominated (Pareto archive), $\mathcal{F}(L^t)$, or dominated, $\mathcal{D}(L^t)$. Thus, hosts are identified for replacement. Should symbionts *no longer* receive any host indexes then this is taken to imply that they were only associated with the worst L_{gap} hosts and they therefore 'die'. The symbiont population size is therefore free to float.

Finally, the role of function 'Best' (line 19) is to return the champion host for evaluation against the test partition. Given the streaming context – or continuous evolution against the stream – all hosts currently in the Pareto archive are evaluated against the current content of the point population. The assumption being that if the Pareto archiving strategy is effective, then the points remaining in the point population should be the most appropriate for prioritizing the champion host. The metric assumed for champion identification is the average class-wise detection rate or:

$$avg.DR = \frac{1}{|C|} \sum_{c \in C} DR_c(l_i) \tag{5}$$

where C is the set of class labels associated with this task and $DR_c(l_i)$ is the detection rate of host l_i relative to class c. Readers are referred to [17, 8] for additional SBB details e.g., instruction set and variation operators.

4 Evaluation

4.1 Diversity versus aging heuristics

Relative to the Pareto archiving framework of Section 3.2, we note that at present a fitness sharing heuristic is used to promote the maintenance of diversity once the number of non-dominated points encounters the constraint

Algorithm 1 Overview of the SBB training algorithm as applied to streaming data τ. S^t denotes the set of training instances associated with the sliding window at generation t; τ denotes the streaming data; $|P|$ and P^t are the size and content of the point population respectively; $|L|$ and L^t are the size and content of the host population respectively;.

```
1:  procedure TRAIN
2:      t = 0                                              ▷ Initialization.
3:      S^t = INITSTREAM(τ(t))
4:      P^t = INITPOINTS(P, S^t)
5:      (L^t, L^t) = INITTEAMS(M)
6:      while t ≤ t_max do
7:          P^t = GENPOINTS(P^t, S^t)                      ▷ Add 'gap' size points
8:          (L^t) = GENTEAMS(L^t)                          ▷ ... and hosts
9:          for all l_i ∈ L^t do
10:             for all p_k ∈ P^t do
11:                 EVALUATE(l_i, p_k)                     ▷ Evaluate fitness
12:             end for
13:         end for
14:         S^{t+1} = SHIFTSTREAM(τ(t + 1))                ▷ Resample
15:         P^{t+1} = SELPOINTS(P^t)
16:         (M^{t+1}) = SELTEAMS(L^t)
17:         t = t + 1
18:     end while
19:     return BEST(L^t, P^t)
20: end procedure
```

imposed by a finite archive. Given that streaming applications are often of a real time nature, letting archive size increase to include all non-dominated points is not a desirable option [2]. Moreover, once a point provides a distinction it could potentially lie in the point population indefinitely i.e., the case of an outlier or mislabelled data. Under streaming applications this is also deemed to be undesirable as the underlying process determining stream content are frequently non-stationary. With this in mind, the fitness sharing heuristic of Equation 3 will be adjusted with the goal of evaluating its role in allowing the GP populations to interact more effectively with the current contents of the stream. To that end, the following three configurations will be explored:

1. Fitness sharing in its original, unmodified format. This case will serve as the base case for attempted optimizations to the fitness sharing heuristic. Hereafter this case is denoted **org**.

2. Fitness sharing scores multiplied by the normalized inverse of point archive ages; that is, $score = score \times (1 - age/age_{max})$. Naturally, this heuristic introduces an age bias, with the motivation being to make older points more likely to be replaced. Hereafter this case is denoted **age**.

3. Points with a fitness sharing score of 0 are considered to be ranked highest; after this, points are taken in the order of highest score first. This case is intended to address the issue of new points which **no** GP individuals are able to correctly classify being unable to enter the point archive. This prevents material that is currently entirely unclassifiable from being presented to the team population as a desirable problem to be solved. Hereafter this case is denoted **zero**.

Since online learning applications require an answer in the "here and now" of a data stream and not in a separately partitioned external environment, testing is performed at periodic intervals during training by extracting the current front-running GP individual and evaluating it separately across the previous interval, and the next (as-yet-unseen) interval. The evaluation metrics used will be the accuracy (total number correct versus the total number of exemplars in the interval) and average detection rate (Eqn (5)) as calculated across the interval. The relative variation between accuracy and average detection rate provide a characterization for how sensitive the resulting classifier is to class imbalance.

4.2 Datasets

Two methods of generating synthetic data for benchmarking streaming classification algorithms are employed:

Table 1: Characterization of benchmarking datasets. Attribute counts appear next to the respective dataset names.

Class	datgen (11)	planar (7)
1	2,615,747	14,055
2	1,801,055	120,167
3	1,643,327	15,778
4	635,629	–
5	304,242	–

1. Dataset Generator[3], a tool for generating benchmark data for data mining applications, is used to generate exemplars representing three separate concepts, C1, C2, and C3. These concepts are then mixed as in [30]: a stream of 7,000,000 total exemplars is divided into "chunks" containing 500,000 exemplars each, and containing representation from each of the three concepts in a tuple written in the form $\langle$%C1, %C2, %C3$\rangle$. The entire stream can then be written as fourteen chunks expressed, in order, as follows: $\langle 100, 0, 0\rangle$ $\langle 100, 0, 0\rangle$ $\langle 100, 0, 0\rangle$ $\langle 90, 10, 0\rangle$ $\langle 80, 10, 0\rangle$ ... $\langle 10, 90, 0\rangle$ $\langle 0, 100, 0\rangle$ $\langle 0, 0, 100\rangle$. The resulting data set will be denoted **datgen**.

   The data is generated using the parameterization set forth in [30]. Five real-valued attributes are specified that are relevant to the class attribute, and a sixth attribute that is irrelevant. Datgen then creates a decision tree-style list of rules that partitions the attribute space into 5 classes based on randomly generated threshold values, and assigns a class label to each of the tree branches. Exemplars are then created by randomly determining attribute values and evaluating them against the generated decision tree in order to assign a class label to the instance.

2. Two hyperplanes of the form $\Sigma_{i=1}^{d} a_i x_i = a_0$ are generated, as in [9], where the values of a_i represents the current state of the stream (initialized randomly), the values of x_i represent the attribute values of a given exemplar, and $d = 10$ represents the chosen dimensionality for the task domain. A stream of 150,000 exemplars is created, and concept drift is parameterized and simulated as follows: Every $N = 1,000$ exemplars, each entry in a vector $s = \{-1, +1\}^d$ (also initialized randomly) is given a 20% chance of inverting its value. The first 5 of 10 non-class attributes in a are then moved at a rate of $t = 10\%$ over the course of the next N exemplars, compounded every generation. This function is summarized in Equation 6 :

$$a_i = a_i + s_i \cdot \frac{a_i \cdot t}{N} \tag{6}$$

   Exemplars are created by generating uniform random values for all non-class attributes. To determine the class label, the values a_0 for each hyperplane are first normalized by summing the rest of the entries in the vector and multiplying by a factor of 1/3 and 2/3 respectively, in order to keep the class distribution from becoming entirely one-sided. The class label is then equal to the number of hyperplanes for which $\Sigma_{i=1}^{d} a_i x_i < a_0$. The resulting dataset will be denoted **planar**.

This provides two basic scenarios for comparing the performance of the heuristics under different styles of concept drift. The first is a discrete case in which two different concepts cannot be interpolated between, and instead only the frequency of their appearance in the surrounding stream changes over time. The second is a continuous case in which two different concepts are related to each other by some randomly drifting function, and a given exemplar may exist on some fuzzy boundary between the two as the stream's make-up slowly drifts from being composed of one concept to another.

In keeping with established practice in generating synthetic data sets for classification, new exemplars are created by randomly choosing values for each attribute and then applying the generated ruleset to determine the exemplar's class attribute. This tends to lead to a natural class imbalance dependent on how much of the attribute space falls under the jurisdiction of each generated rule. The datasets are summarized in Table 1.

4.3 Parameterization

There are two parameters specific to the interface between SBB and the sliding window:

Maximum number of generations, t_{max}: This defines the number of generations conducted while making a single pass through the data set (Section 4.1), and is held constant across all datasets. The impact of this is that new

[3]Gabor Melli. The datgen Dataset Generator. http://www.datasetgenerator.com/

Table 2: SBB parameterization.

	Description	Value
P_{size}	Point population size.	120
M_{size}	Team population size.	120
t_{max}	Number of generations.	30 000
p_d	Probability of learner deletion.	0.7
p_a	Probability of learner addition.	0.7
μ_a	Probability of action mutation.	0.1
ω	Maximum team size.	20
P_{gap}	Point generation gap.	20
M_{gap}	Team generation gap.	60

training instances become eligible at the rate of ≈ 233 per generation under datgen and ≈ 5 per generation under planar (i.e., total training instance count $\div\ t_{max}$).

Window size, w, for the sliding window: This is set at 5% of the total dataset size, or 350,000 records for datgen and 7,500 for planar. A larger window size allows more data to be sampled for possible inclusion in the archive at a time (and thus more representation of the current concept distribution of the stream), at the cost of increased storage costs.

SBB parameters: are the same as the original study on Pareto archiving [2], and shown for convenience in Table 2).

In each case a total of 50 runs are performed per experiment.

4.4 Results

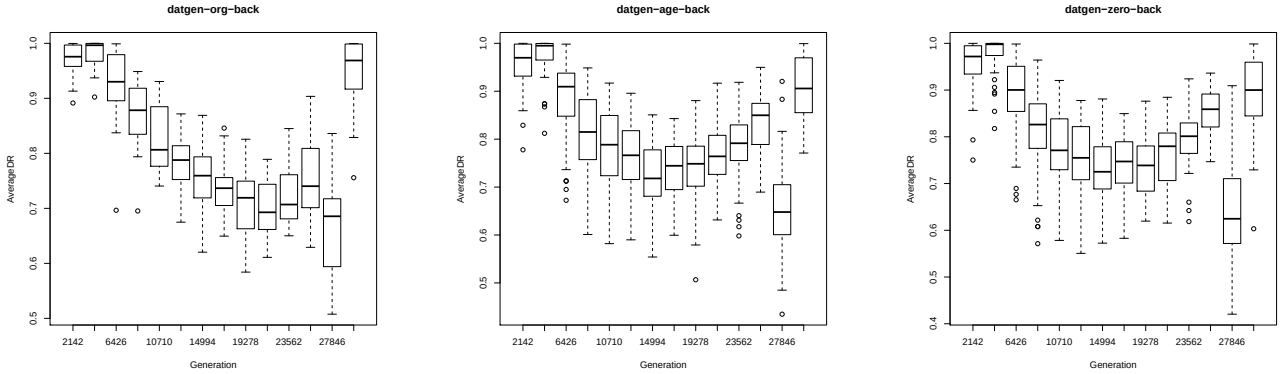
As outlined in Section 4.1, a total of three scenarios are considered across the **datgen** and **planar** data sets: 1) Unmodified fitness sharing scores or the **org** configuration. 2) Fitness sharing scores multiplied by inverse normalized exemplar age, or the **age** configuration. 3) Fitness sharing scores left unmodified, except scores of 0 are changed to ∞, or the **zero** configuration.

The average detection rate metrics, as calculated across periodic intervals in the stream, are shown in Figure 1 for the datgen data set, and in Figure 2 for the planar dataset.

In the base case **org** for the **datgen** stream, performance hovers near unity for the first two intervals while the underlying concept C1 does not change, then begins to steadily decline as data points from the second concept C2 are gradually introduced. The decline in performance continues until the tenth chunk, at which point more than two-thirds of the stream is composed of representatives of C2, and only then does the algorithm begin adapting to the change in the underlying environment. Interestingly, the next-to-last chunk of data composed entirely of C2 exemplars caused the performance of the algorithm to drop the lowest, but the team population is able to easily adjust to a brand new concept C3 when it abruptly takes over the stream (last chunk of the sequence). Such behaviour would seem to imply a weakness in the current GP model for adapting to change spread out over a significant period of time, at least in the case of discrete concept representation. Conversely, sudden change appears to result in a more effective retraining and replacement of GP classifiers.

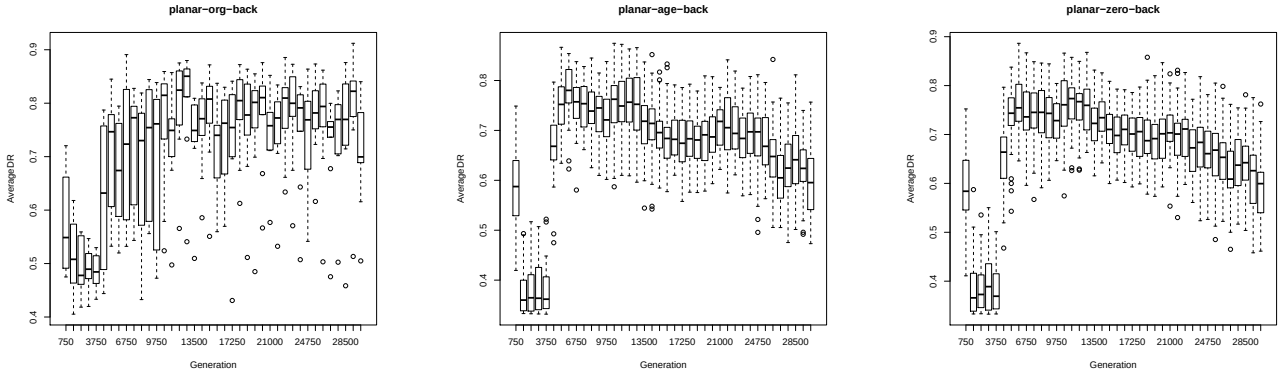
The other experiments **age** and **zero** in the **datgen** environment exhibit similar behaviour but appear to be able to start compensating for the gradual change in concept earlier in the stream. Thus, the quartile statistic is better at remaining above an average detection rate of 70% than under **org**. Of the three, the **age** case appears to adapt the quickest. This configuration, as described in Section 4.1, was intended to punish points that have spent more time in the archive – to the extent that the oldest point is deterministically removed at each generation.

In all cases for the **planar** stream, performance in terms of average detection rate appears to suffer initially from a pathological case of class imbalance. This is corroborated by the high accuracy metrics reported for that portion of the stream (Figure 3), and can be verified by looking at the class distribution of the point archive from an example run (not shown). In this case, the original diversity based heuristic appears to resist the incremental variation in the stream content better than either the explicitly age or 'disengaged point' biases or **age** and **zero** respectively. This is particularly true under the balanced detection rate metric of Figure 2. Note the different



(a) org (b) age (c) zero

Figure 1: Average Detection Rate across previous chunks during training on the stream, using the *datgen* data set.



(a) org (b) age (c) zero

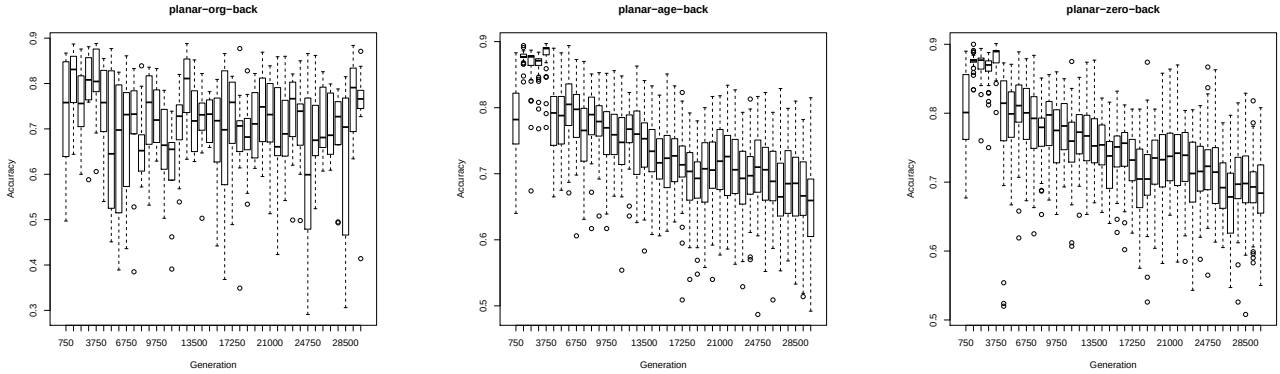
Figure 2: Average Detection Rate across previous chunks during training on the stream, using the *planar* data set.

scales under the accuracy metric in Figure 3. In this case there is much more variation in the spread of the original heuristic (e.g., 1st quartile extends down to 50%) whereas the **age** and **zero** heuristics demonstrate less variance. However, in either metric, median performance never descends below 70% under **org** whereas both the alternative heuristics fail to do so.

The magnitude of the effect of the different fitness sharing heuristics on the archive policy's age bias can be clearly seen in Figure 4. The **org** heuristic constantly churns the contents of the point archive, whereas both **age** and **zero** provide an additional level of granularity to the score values that allow the archive to distinguish between points enough to single some out for more long-term archiving.

5 Conclusion

Pareto archiving and the development of GP teams through symbiosis are proposed as a general framework for supporting the multiple factors deemed to be of significance to continuous evolution in non-stationary environments. Specifically, the SBB algorithm supports task decomposition through speciation and bidding, whereas evolvability is addressed through the use of independent representations and variation operators for teams and programs. Pareto archiving provides a formal framework for retaining specific learners (GP teams) and the corresponding data points, or a memory mechanism. This study introduces two multi-class benchmarks of importance to streaming under non-stationary tasks. Specifically, the 'datagen' task assumes stepped changes to the underlying process whereas 'planar' adopts a continuously varying process. The resulting benchmark study indicates that diversity is more effective under the continuously varying process. Conversely, introducing an additional age bias that deterministically retires the oldest non-dominated learners / points is more effective in the non-stationary task with stepped changes.



(a) org (b) age (c) zero

Figure 3: Accuracy across previous chunks during training on the stream, using the *planar* data set.

Figure 4: Average age of individuals in the point archive from single runs of experiments in the *planar* environment.

In the case of future research we note that GP individuals are currently only evaluated with respect to a single exemplar. Conversely, data structures such as tapped delay lines [27] and evolved sliding window parameterizations [28] would further enhance the capability of GP individuals to characterize the time varying nature of the underlying task, and possibly introduce a method of dealing with cyclical concept drift behaviour which has not been examined here. We are also interested in the use of GP within active learning environments to perform change detection without the use of labeled data. This would imply that only when changes are detected would labels be requested.

6 Acknowledgements

A. Atwater was supported in part by an NSERC CGS-M scholarship.

References

[1] H. Abdulsalam, D. B. Skillicorn, and P. Martin. Classification using streaming random forests. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):22–36, 2011.

[2] A. Atwater, M. I. Heywood, and A. N. Zincir-Heywood. Gp under streaming data constraints: A case for Pareto archiving? In *ACM Genetic and Evolutionary Computation Conference*, pages 703–710, 2012.

[3] P. Bruneau, F. Picarougne, and M. Gelgon. Incremental semi-supervised clustering in a data stream with a flock of agents. In *IEEE Congress on Evolutionary Computation*, pages 3067–3074, 2009.

[4] J. Cartlidge and D. Ait-Boudaoud. Autonomous virulence adaptation improves coevolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):215–229, 2011.

[5] W. Cedeno and V. R. Vemuri. On the use of niching for dynamic landscapes. In *IEEE Congress on Evolutionary Computation*, pages 361–366, 1997.

[6] E. D. de Jong. A monolithic archive for pareto-coevolution. *Evolutionary Computation*, 15(1):61–93, 2007.

[7] I. Dempsey, M. O'Neill, and Brabazon A. *Foundations in Grammatical Evolution for Dynamic Environments*, volume 194 of *Studies in Computational Intelligence*. Springer, 2009.

[8] J. A. Doucette, A. R. McIntyre, P. Lichodzijewski, and M. I. Heywood. Symbolic coevolutionary genetic programming: A benchmarking study under large attribute spaces. *Genetic Programming and Evolvable Machines*, 13:to appear, 2012.

[9] W. Fan, Y. Huang, H. Wang, and P. S. Yu. Active mining of data streams. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, pages 457–461, 2004.

[10] S. G. Ficici and J. Pollack. Pareto optimality in coevolutionary learning. In *European Conference on Advances in Artificial Life*, pages 316–325, 2001.

[11] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving from Nature*, pages 312–321, 1994.

[12] A. Ghosh, S. Tstutsui, and H. Tanaka. Function optimization in non-stationary environment using steady state genetic algorithms with aging of individuals. In *IEEE Congress on Evolutionary Computation*, pages 666–671, 1998.

[13] J. J. Greffenstette. Genetic algorithms for changing environments. In *Proceedings of Parallel Problem Solving from Nature*, pages 137–144, 1992.

[14] M. Kotanchek, G. Smits, and E. Vladislavleva. Exploiting trustable models via pareto GP for targeted data collection. In *Genetic Programming Theory and Practice VI*, pages 145–162. Springer, 2009.

[15] M. Lemczyk. Pareto-cevolutionary genetic programming classifier. Master's thesis, Faculty of Computer Science, Dalhousie University, 2006. http://web.cs.dal.ca/~mheywood/Thesis/MCS.html.

[16] P. Lichodzijewski and M. I. Heywood. Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In *ACM Genetic and Evolutionary Computation Conference*, pages 464–471, 2007.

[17] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with symbiotic bid-based genetic programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[18] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under gp. In *ACM Genetic and Evolutionary Computation Conference*, pages 853–860, 2010.

[19] A. R. McIntyre and M. I. Heywood. Cooperative problem decomposition in pareto competitive classifier models of coevolution. In *European Conference on Genetic Programming*, pages 289–300, 2008.

[20] R. W. Morrison. *Designing evolutionary algorithms for dynamic environments*. Springer, 2004.

[21] J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In *ACM Genetic and Evolutionary Computation Conference*, pages 493–500, 2001.

[22] A. Orriols-Puig, J. Casillas, and E. Bernado-Mansilla. First approach toward on-line evolution of association rules with learning classifier systems. In *ACM Genetic and Evolutionary Computation Conference*, pages 2031–2038, 2008.

[23] V. Ramos and A. Abraham. Swarms on continuous data. In *IEEE Congress on Evolutionary Computation*, pages 1370–1375, 2003.

[24] J. Reisinger, K. O. Stanley, and R. Miikkulainen. Towards and empirical measure of evolvability. In *ACM Genetic and Evolutionary Computation Conference*, pages 257–264, 2005.

[25] D. Saad, editor. *On-line learning in neural networks*. Cambridge University Press, 1998.

[26] H. A. Simon. *The sciences of the artificial*. MIT Press, 2nd edition, 1996.

[27] S. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patters: An example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, 2005.

[28] 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, 2007.

[29] S. X. Wu and W. Banzhaf. Rethinking multilevel selection in genetic programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 1403–1410, 2011.

[30] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 40(6):1607–1621, 2010.