A New Wave: A Dynamic Approach to Genetic Programming

David Medernach
BDS Group
CSIS Department
University of Limerick
david.medernach@ul.ie

Jeannie Fitzgerald
BDS Group
CSIS Department
University of Limerick
jeannie.fitzgerald@ul.ie

R. Muhammad Atif Azad
BDS Group
CSIS Department
University of Limerick
atif.azad@ul.ie

Conor Ryan
BDS Group
CSIS Department
University of Limerick
conor.ryan@ul.ie

ABSTRACT

Wave is a novel form of semantic genetic programming which operates by optimising the residual errors of a succession of short genetic programming runs, and then producing a cumulative solution. These short genetic programming runs are called periods, and they have heterogeneous parameters. In this paper we leverage the potential of Wave's heterogeneity to simulate a dynamic evolutionary environment by incorporating self adaptive parameters together with an innovative approach to population renewal. We conduct an empirical study comparing this new approach with multiple linear regression (MLR) as well as several evolutionary computation (EC) methods including the well known geometric semantic genetic programming (GSGP) together with several other optimised Wave techniques. The results of our investigation show that the dynamic Wave algorithm delivers consistently equal or better performance than Standard GP (both with or without linear scaling), achieves testing fitness equal or better than multiple linear regression, and performs significantly better than GSGP on five of the six problems studied.

CCS Concepts

•**Computing methodologies** → **Genetic programming;** *Genetic algorithms;*

Keywords

Natural Selection, Semantic GP, Genetic Programming, Symbolic Regression, Self-adaptation, ensembles, residuals

1. INTRODUCTION

A significant proportion of recent research on genetic programming (GP) [11] has focused on designing genetic operators that are aware of the *semantics* of the evolving individuals. Wave [13] is a form of GP which shares some similarities with semantic GP methods such as Sequential Symbolic Regression (SSR) [19]. As with SSR, rather than using a single run, Wave employs a succession of runs that produce a cumulative solution by training on the residual errors of previous runs. In Wave [14], these runs are called *periods*. The final output of Wave then is a *joint solution* which is an algebraic sum of the best evolved solutions from different periods. This idea of restarting runs in the form of a period to optimise the residual is necessitated by the empirical observation that the rate of fitness gain in GP, typically, is at its highest during the early generations and levels off towards the end of a run; moreover, as the fitness gains recede, *bloat* co-occurs: bloat is an increase in the average size of the population without a significant fitness gain. In contrast, Wave starts another period when it detects that the rate of improvement has slowed down. Since we optimise the residuals in each successive period, the previously discovered semantics of the developing joint-solution guide further evolution. Therefore, Wave, like SSR, belongs to the class of semantics-guided GP or simply semantic GP.

Wave also employs *heterogeneous configurations* across its periods, for example, different periods may span a different number of generations depending upon how well a period has been progressing, and may or may not choose to use *linear scaling* [10] to optimise the slope and the intercept of the evolving functions. Medernach et al. [13] have shown that Wave is an effective method for solving regression problems and experimented with various Wave setups that used different population sizes. They reported that the most successful configuration used a population of 500 individuals and alternated between periods with and without linear scaling, with linear scaling activated at the first period.

While empirical results necessitate the Wave approach, its design is strongly rooted in lessons from biological evolution. One such idea is the *punctuated equilibrium* [7] which observes that evolution in the living world is not linear, rather it alternates between a succession of periods of *stasis* and

rapid change. Wave seeks to emulate this dynamic evolutionary environment by simulating periods of rapid change using mechanisms inspired by evolutionary biology. These mechanisms are *saltationism* [3] where a saltation is defined as non gradual modifications of the genotype – *macromutations*, and ecological change [15]; both saltationism and ecological change may promote rapid speciation. Wave depicts saltationism in an EC context by starting each new period with a new population and by modifying the objective function at the beginning of each Wave period to model an environmental change event. The reintialisation of population and a changing objective function across periods in Wave also reconcile with an idea developed by McClintock [12] which suggests that environmentally induced stressful conditions may trigger an adaptive response, leading to massive genetic variations in the genomes of plants. In [9], the authors suggest that these variations will be triggered by epigenetic mechanisms responsive to the stress, and that this trigger response mechanism may be fairly common in biological evolution.

We hypothesise that modifying Wave *further* such that to bring it closer to the themes which inspired it and are successful in biological evolution may allow us to simulate a dynamic evolutionary environment which may yield improved performance.

In the *punctuated equilibrium* model of evolutionary biology, *stasis* is defined as a period of little or no evolutionary change in a species. Effective detection of stasis is critical in Wave so as to conserve the computational effort by stopping a period. Once a period stops, the best individual from that period may be incorporated into the joint solution if it improves the joint solution in some fashion. However, if a period stops *too early*, selection may not have sufficient time to effectively optimise, whereas if the period stops *too late*, it only wastes computation cycles while the population bloats and potentially *over-fits* the training data. Thus, in this paper, we explore a smart, adaptive stopping criteria for the Wave periods using a *validation set*.

In fact, we take a number of self-adaptive measures in this paper to extend Wave; for this we propose that Wave *automatically* selects the number of periods for each run, the duration of each period and whether or not to use linear scaling in a given period.

Furthermore, because each period starts with a fresh population (like a new run), the previously evolved genetic material gets inaccessible; we hypothesise that it may, therefore, inhibit the establishment of mechanisms analogous to *exaptation* [8]: exaptation occurs when features that now enhance fitness were not originally built by natural selection for their current role. Exaptation eases the evolution of complexity in the living organisms. To facilitate exaptation, in this paper, we propose to only partially renew the population at the beginning of each period, thus retaining some of the evolved genetic material.

2. BACKGROUND

In this section, after briefly describing the original Wave implementation, we provide an outline of some of the most important and recent work on the relevant topics, that is, population renewal strategies and the use of a validation dataset.

2.1 Original Wave

Any period, optimizing over a current target set t of size N, is considered to be *successful* if the best trained individual resulting from the period generates a function f with a RMSE at least better than a neutral individual represented by 0, as below.

$$\sqrt{\sum_{i=1}^{N}(f_i - t_i)^2} < \sqrt{\sum_{i=1}^{N}(t_i - 0)^2} \quad (1)$$

When a *successful* period terminates, the new target data-set t' for the next period is defined such that $t' = t - f$. However, if the period is not successful, the new period starts with an unchanged target $t' = t$. Therefore, $f_{joint} = \sum_{j=1}^{n}(f_j)$ where f_{joint} is the joint solution of a Wave run, n is the number of successful periods and f_j is the function produced by the jth successful period.

2.2 Population Renewal

Regular generation of completely new individuals is a technique used by some evolutionary algorithms to avoid premature convergence of the population to a local optimum in the fitness landscape. In [21], for example, individuals created at different generations coexist. The selection then compares only individuals of similar age; this prevents the older individuals, who have had more generations to evolve, from dominating the younger individuals. In Wave, the entire population is replaced at the beginning of every new successful period. Thus, the question of the disadvantage of certain individuals does not arise across successful period. However, we believe that in Wave a new period can utilise some of the individuals from the previous period without having to worry about the older individuals dominating the fresh population: this is because the objective function changes across successful periods and thus the older individuals will also have to adapt according to the new data just like the new individuals. Thus, unlike in [21], Wave does not require special mechanisms to *protect* the freshly generated individuals from the fitness advantage that the older individuals from the previous period might have enjoyed if the objective function had not changed. Instead, the research question is whether the older individuals can still be useful in accelerating evolution alongside a fresh population, and thus exhibit exaptation.

2.3 Validation Sets and Smart Stopping

A common application of a validation dataset in machine learning involves using as a control system a subset of the data which is not used to train the model. The performance of a learning system on this validation dataset may provide a useful estimate of the likely performance on *out of sample* data, that is, how well the learnt model may generalise beyond the training data [20]. The use of validation sets can reduce over-fitting [2], detect *stasis*, that is, a lack of improvement on the validation set, and can also double up as an early stopping mechanism [5][18] in order to start a new period. Here, we propose to use a validation set to decide when to terminate the current period and also as to whether we must integrate the best individual of the current period into the joint solution.

3. PROPOSED METHOD

Figure 1 depicts the logic of both the original and enhanced versions of the Wave system. In this section, we describe various components of the enhanced paradigm. Some of these techniques such as smart stopping of periods apply to all Wave configurations, whereas others are used only in specific setups. Table 2 outlines these details.

3.1 Partial Population Renewal

Previously, the entire population was renewed at the beginning of a new period. In this study we only renew 80% of the original population[1]; the 20% of individuals that are not replaced are randomly selected. As discussed earlier, the fresh individuals at the start of a new period do not necessarily suffer a disadvantage when pitted against the evolved peers from the previous period: this is because the objective function has changed[2]. Unless there is a strong correlation between the new and old objective functions, the old individuals do not carry an advantage[3]. Therefore, unlike the approach taken in [21], we do not require specialised mechanisms such as age of the genotype to preserve the disadvantaged individuals. Of course, pre-existing individuals who may have had good fitness in the previous period, may initially be unable to solve the redrafted problem. However, we believe that maintaining this genetic material into the gene pool offers the possibility of the emergence of phenomena such as exaptation or reuse of existing modules, while at the same time allowing the new genetic material to flourish.

3.2 Smart Stopping of Wave Periods

Previously, in [13], a period was terminated when the Wave system determined that the period was not significantly improving the fitness any longer. – if the improvement, in terms of the best training fitness, over the two most recent generations was less than 0.5% of the improvement over the three generations previous to the two most recent generations.

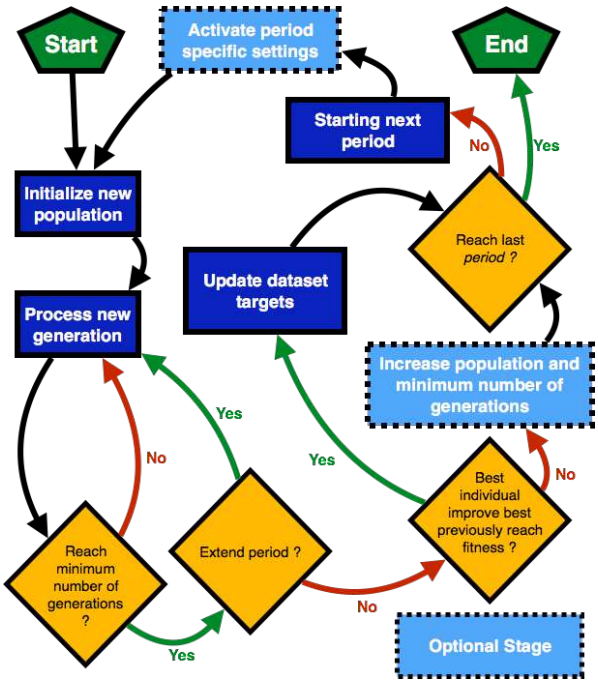
In this work we investigate a new strategy whereby, in every generation, we also compute the fitness of the best trained individual on a validation set. Suppose O(f,d) represents the objective fitness (or fitness value) of an individual function f on a dataset d, then $O(f_p(g), v_p)$ is the objective fitness of the individual $f_p(g)$ evaluated over the validation dataset v_p; note $f_p(g)$ is the best trained individual at generation g of the period p. Considering we are minimizing the fitness, we then terminate a period if $O(f_p(g), v_p) > O(f_p(g-1), v_p)$. In other words, we stop a period *whenever* the validation fitness degrades. Granted, that the validation fitness may oscillate in future, if the period is allowed to progress; however, our preliminary investigations revealed that waiting for that to happen did not improve scores on the unseen test data sets.

Often, in our experience, especially when linear scaling is enabled, the validation fitness does not change over many generations, that is, $O(f_p(g), v_p) = O(f_p(g-1), v_p)$; this rep-

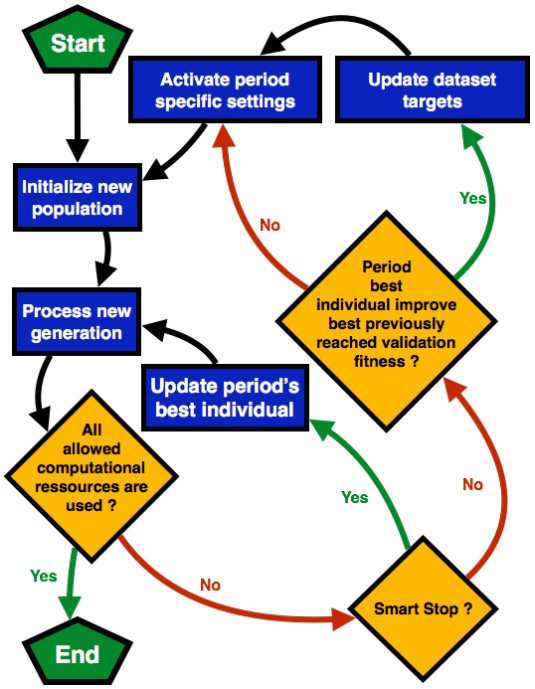
[1]First, we explored replacing only 50% of the population; however, the population converged very quickly without improving the results.

[2]Renewing the target set as $t' = t - f$ akins here a stressful ecological change.

[3]A possible extension of this work is to measure this correlation and adjust the relative proportion of new and old population members accordingly.



(a) Original Wave



(b) Enhanced Wave

Figure 1: Flow of Wave algorithm.

resents stasis. We decide to halt the current period if stasis continues for more than 5 consecutive generations; however, if at any time, an individual that is *eligible* to be a part of the joint solution is produced, we allow the current period to have another 25 generations[4] of stasis[5]. The eligible individual is the one that produces the best validation fitness thus far, across all the periods. Formally,

$$O(f_p(g), v_p) < \min_{i=0\ldots p-1} O(f_i, v_i) \quad (2)$$

.

The same criteria is reapplied, as below, at the end of the period to check whether this period has produced *any* solution that is eligible to be a part of the joint solution:

$$O(f_p, v_p) < \min_{i=0\ldots p-1} O(f_i, v_i) \quad (3)$$

where f_p is the best trained individual at any generation during the current period p that also has the best validation fitness. Due to the stopping conditions described above, this individual is always present in the penultimate generation of the period. Note, the eligibility criteria to get into the joint solution described here is different from that in the previous implementation of Wave described in section 2.1.

3.3 Adaptive Linear Scaling

Previously, the most effective Wave configuration involved alternating between periods with and without linear scaling. However, our exploratory runs indicate that the success rate of periods with and without linear scaling is not uniform. We also noticed that the success rate of linear scaling is proportionally much higher in the first period than in the last period. Moreover, all this is highly dependent on the problem under consideration. So, in this work, we have chosen not to alternate between periods with and without linear scaling. Instead, at the beginning of each period, the system chooses randomly between these two modes based on their success rate in preceding periods. To achieve this the probability of activating linear scaling for a given period is $P = \frac{r_{LS}}{r_{LS}+r_{\neg LS}}$ where $r_{LS} = \frac{LS_{suc}+1}{LS_{tot}+1}$ is the approximate success rate of periods using linear scaling; LS_{tot} is the total number of periods that used linear scaling *thus far* and LS_{suc} is the number of successful periods that used linear scaling; $r_{\neg LS}$ is the approximate success rate of period not using linear scaling computed the same way. It is important to note that these counts of various kinds of periods stabilise as the number of periods grow thus giving a more reliable measure of the probability of activating linear scaling.

4. EXPERIMENTS

4.1 Benchmarks

We compare Wave with both EC based and non-EC based machine learning methods. We will use standard GP, both with and without linear scaling to benchmark both the computational cost of evolution and accuracy of the resulting models. While we use GSGP [16] and Multiple Linear Regression (MLR) solely to benchmark the accuracy of the achieved results. This is because GSGP caches evaluations from the first generation and does not evaluate new material afresh; instead, this work does not cache evaluations yet and therefore can not compare with GSGP. Also, MLR is well known to be far cheaper than GP yet has recently been proposed as a tough benchmark for GP to beat [1].

4.2 Problem Suite

For this study we have used three multi-dimensional datasets from the UCI Machine learning repository [2] together with two mathematical functions. The following datasets from the UCI repository are used:

- **Concrete Strength** where the objective is to predict the compressive strength of concrete and data-set includes 1030 instances each having 8 inputs.
- **Yacht** where the objective is to predict the hydrodynamic performances of a yacht, and data-set includes 308 instances each with 7 inputs.
- **Powerplant** where the task is to predict the net hourly electrical energy output of a power plant and data-set includes 9568 instances and 4 inputs.

The two mathematical functions chosen are:

- **Poly-10 [22]** $y = x1 * x2 + x3 * x4 + x5 * x6 + x1 * x7 * x9 + x3 * x6 * x10$ and
- **Div-5 [23]** $y = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$.

For each of the two mathematical functions, 500 data points in the range $[0; 1]$ are randomly generated. For each run we randomly split the given data-set into three subsets of equal size for training, validation and testing purposes, that is, $N_{training} = N_{validation} = N_{testing}$. For the benchmark algorithms, because they do not use periods and therefore can not use a validation set the way we do here[6], the instances in the validation set are added to the training set, the numbers of testing instances remains of course the same between the benchmarks and Wave.

Previously, in [14] the Wave system did worse than standard GP on the Yacht problem. As Wave had done better on all of the other datasets we are interested in understanding the cause of the difference in performance. We suspect that the relatively small size of the Yacht dataset (308 data points) may have contributed to higher over-fitting. In order to examine this hypotheis we select the Concrete Strength dataset on which Wave performed well and create a smaller version of this dataset to use as a surrogate. Thus, we conduct separate runs on smaller training sets from the **Concrete Strength** problem where the $N_{training} = N_{validation} = 103 \approx 308/3$. The remaining points make the test set such that $N_{testing} = 824 \approx 1030 - 206$. We will refer to this particular configuration as the **Small Concrete** configuration from now on.

4.3 Common Parameters

The parameters described in Table 1 have been chosen to be the same as in [14] and are consistently applied across all Wave and other GP benchmark experiments. We process 45 runs for each EC configuration for each dataset. Because of

[4]These choices appear adhoc but they are based on some empirical success; however, they are not necessarily optimal.
[5]Note, we still stop the period at any time the validation fitness degrades.

[6]Exploratory experiments also indicated that standard GP does not benefit from a validation dataset using one third of the total available datas on the datasets used in this paper.

its low computational cost we performed 100 MLR runs on each dataset.

Table 1: GP Parameters.

Parameter	Value
Population	500 individuals
Replacement Strategy	Generational
Operator Probabilities	Xover: 0.9; Point mutation: 0.1
Tournament Size	10[a]
Max. depth	17
Max. size	100
Functions set	+,−,×,÷(Protected division)
Terminal set	Inputs and constants -1.0, -0.5, 0.0, 0.5 & 1.0
Fitness	RMSE
Initialisation	Ramped half & half
Max. initial depth	8
Mutation Step (GSGP Only)	0.1[b]

[a] A relatively high tournament size but recently successfully used in [6] and [17]; we kept the default tournament size of 3 on GSGP.
[b] This is the default mutation step in the implementation proposed in [4].

4.4 Experimental Configurations

In this paper we study two Wave configurations using both adaptive linear scaling and smart stopping: the first configuration does not use any optional settings; the second one uses partial population renewal. All of these configurations and benchmarks are described in Table 2.

Table 2: Different Wave and benchmarks configurations.

Name	**Wave**	**Renew** [a]	**LS**
GP with linear scaling:			
GP:LS	Off	NA	On
Standard GP:			
GP	Off	NA	Off
Multiple linear regression:			
MLR	NA	NA	NA
Geometric Semantic GP as per [4]:			
GSGP	NA	NA	NA
Wave:			
Wave	On	100	Alt[b]
Wave with partial population renewal:			
Wave:PPR	On	80	Alt[b]

[a] Proportion of population renewed at the beginning of each period.
[b] Alternation between LS and non-LS periods, based on LS and non LS success rate.

4.5 Performance Metrics

It is not relevant to compare the Wave approach with conventional GP simply by measuring the performance of each on the same number of generations. This is because the population is regularly renewed in Wave so that individuals do not have the same opportunity to bloat as they do in conventional GP. Consequently, each generation is generally less computationally expensive to process with Wave. Therefore, we believe that it would not be equitable to take an approach of terminating runs after the same number of generations for each method, nor to compare statistics measured every generation.

We choose to use the total cost C of computed nodes as a measure of computational cost of a run. Typically, the computational cost of a Wave run will be:

$$C = \sum_{g=1}^{g_{tot}} (\sum_{i=1}^{pop} N_{training} \times S_{I_{gi}}) + N_{validation} \times S_{BI_g} \quad (4)$$

where *pop* (population size) is 500 individuals, $S_{I_{gi}}$ is the size of the ith individual I_{gi} of the generation g and $S_{BI_{gi}}$ is the size of the best individual at generation g instead, for classic GP:

$$C = \sum_{g=1}^{g_{tot}} \sum_{i=1}^{pop} N \times S_{I_{gi}} \quad (5)$$

where N is the size of the full data set. To ensure equity of resources allocated to different systems, we stop a run when its computational cost reaches a maximum value C_{max}. We chose[7] :

$$C_{max} = k \times (N_{validation} + N_{training}) \quad (6)$$

where k is a factor chosen so that a run of classic GP with a population of 500 individuals without linear scaling continues for approximately 100 generations. Here we used $k = 4000000$ for **Concrete** and **Small Concrete** and $k = 2000000$ for **Yacht**, **powerplant**, **Div-5** and **Poly-10**. Those values have been chosen to roughly bring standard GP close to 100 generations, except on Powerplant where we choose a smaller value to cater for its high computational cost due to the size of the dataset (9568 instances). On all datasets and with all parameters we report median testing fitness on a 95% confidence interval in Figure 2.

We observe the statistics at intervals of $C_{max}/100$ and we call these intevals *steps* from now on. Nevertheless we only report a single testing value at every step for MLR and GSGP. Of course MLR is not an EC method, therefore it was not possible to report gradual progress. Also, because, as stated in [4], GSGP cached node evaluations, we could not come up with an identical budget for GSGP. However, so as not to put it to a disadvantage, we choose a tougher benchmark from GSGP: we benchmark the Wave results against the *best* test fitness *ever* achieved by GSGP during 2000 generations. Notice, this guards against any over-fitting that GSGP might have experienced and therefore, avoids the pitfall of choosing an inopportune time to report the testing fitness. In fact, as the results show, GSGP mostly achieved its best test fitness well before reaching the end of run; while there is no guarantee that testing fitness can not improve again, conventional wisdom suggests that overfitting is likelier if we extend the GSGP runs any further.

5. RESULTS AND DISCUSSION

5.1 Population Renewal

To study the effects of our partial population renewal setting, Wave:PPR, we report the earliest period at which the ancestors of the last eligible individual were created. As shown in Table 3, a majority of the last eligible individuals added as part of the joint solution have an ancestor created in the first period. Considering the number of periods reported in Table 4 and that we renew 80% of the individuals each period, this result was not obvious. We suspect that this is possibly partly due the relatively large tournaments used, but it also indicates that the genetic material of individuals evolved to address the first objective function still plays a role later on when the objective function has been

[7] $N_{validation} + N_{training}$ is constant on the same dataset for all possible configurations since the benchmarks that do not use validation dataset, append instances of the validation dataset to the training dataset.

modified. This may indicates successful events of exaptation.

Table 3: Proportion of last eligible individuals included in the joint solution who have an ancestor that was created in the first period.

Concrete Strengh	Small Concrete	Yacht	Powerplant	Div-5	poly-10
73%	76%	62%	73%	77%	58%

5.2 Self-adaptation

Looking at Table 4, we see that the number of periods executed and the proportion of them which use linear scaling varies greatly depending on the problem. The use of smart stopping of periods and adaptive linear scaling allows Wave to automatically configure itself depending on the problem.

Table 4: Average number and success rate of periods calculated with and without LS for Wave:PPR.

Problem	Total Periods [a]	with LS [a]	w/o LS [a]
Concrete Strength	52 (57%)	24 (55%)	29 (59%)
Small Concrete	62 (35%)	29 (33%)	33 (36%)
Powerplant	12 (80%)	06 (75%)	06 (91%)
Yacht	40 (39%)	22 (39%)	18 (38%)
Div-5	26 (45%)	22 (56%)	04 (00%)
Poly-10	74 (25%)	49 (29%)	25 (14%)

[a] Number of periods followed by their success rate in brackets.

5.3 Overfitting

As it can be seen by comparing Figures 2a and 2b, the size of the training dataset influences over-fitting. This assumption is reinforced by the absence of over-fitting in Figure 2e where the size of the dataset used is relatively large at 9568 data-points. However, Figures 2c and 2d show that factors other than just the data sizes also affect overfitting; these figures correspond to two datasets, Div-5 and Poly-10, that are identical in size and result from sampling mathematical functions without adding noise and only one them, Poly-10, generate over-fitting.

5.4 Computational Cost

Wave:PPR is the only method tested whose performance either exceeds or equals that of classic GP methods across all datasets at every step. Therefore, Wave:PPR outperforms standard GP both with and without linear scaling. Wave:PPR also produces the smallest joint solutions among Wave settings as depicted in Figure 3.

5.5 Performances

Because of the nature of GSGP and MLR it was not possible to do a step by step comparison with Wave. From step 40 onwards Wave:PPR performs as well as MLR on the Powerplant problem and outperforms MLR on all the other datasets except Small Concrete. The best median testing fitness of best individuals is reached by GSGP well before the latest generation on 4 out of 6 datasets (Yacht: gen 1; Small Concrete: gen 1112; Div-5: gen 600; Poly-10: gen 472), so it is very unlikely that GSGP could improve its result with more computational resources (higher number of generations) on those datasets. From step 10 onwards Wave:PPR performs as well as or better than GSGP on those four problems. It is hard to conclude on the two remaining problems because both Wave:PPR and GSGP reached their lowest median testing fitness at the end of the simulation which means both of them could probably improve their results with more computational resources. But we note that Wave:PPR outperforms GSGP from step 10 onwards while on powerplant the final Wave:PPR testing fitness is statistically equivalent to GSGP reported results. It is also worth noting, that the final testing fitness with Wave:PPR is better than or equal to the best results available (the lowest point of the fitness curve - therefore non-potentially damaged by over-fitting) in a previous investigation by Medernach et al. [14]; although, the previous study has some experimental differences in terms of computational cost with the work reported here.

6. CONCLUSIONS

In this paper, we present several improvements to the previously proposed Wave GP method. Based on the results obtained, the proposed changes make Wave more efficient on the given problems and allow it to adapt to their characteristics. In particular, Wave successfully demonstrates *exaptation* by utilising the older individuals evolved during the previous periods to train against a new objective function; this approach, that we termed as Wave:PPR, seems robust as only multiple linear regression (MLR) performs better than Wave:PPR and that too on one of the problems only. Notice, MLR has been flagged as a tough benchmark for standard GP in recent literature. Additionally, the use of a validation set has improved Wave of ability detect periods of stasis.

7. REFERENCES

[1] Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. Multiple regression genetic programming. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 879–886. ACM, 2014.

[2] Kevin Bache and Moshe Lichman. Uci machine learning repository, 2013.

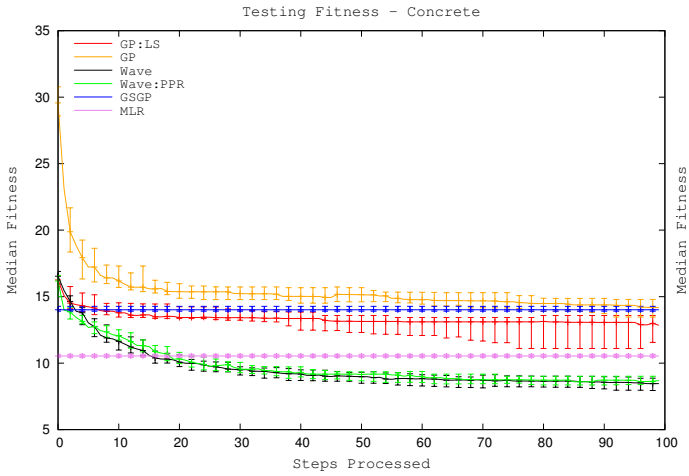
[3] Daniel G Blackburn. Saltationist and punctuated equilibrium models for the evolution of viviparity and placentation. *Journal of Theoretical Biology*, 174(2):199–216, 1995.

[4] Mauro Castelli, Sara Silva, and Leonardo Vanneschi. A c++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines*, 16(1):73–81, 2014.

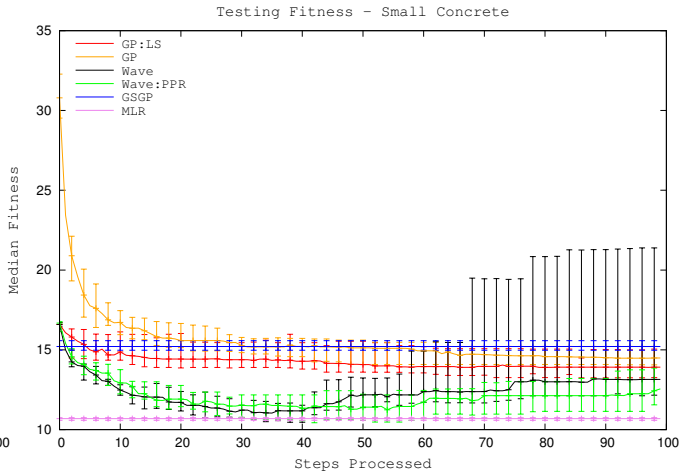
[5] Jeannie Fitzgerald and Conor Ryan. Validation sets for evolutionary curtailment with improved generalisation. In *Convergence and Hybrid Information Technology*, pages 282–289. Springer, 2011.

[6] Ivo Goncalves and Sara Silva. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 73–84, Vienna, Austria, 3-5 April 2013. Springer Verlag.

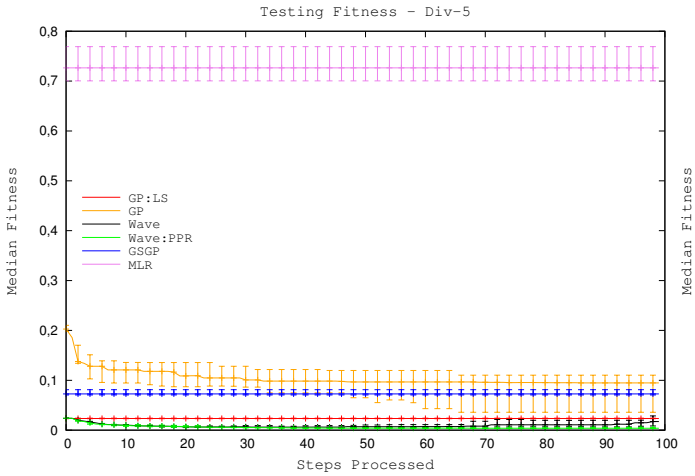
[7] Niles Eldredge-Stephen Jay Gould. Punctuated equilibria: an alternative to phyletic gradualism. 1972.



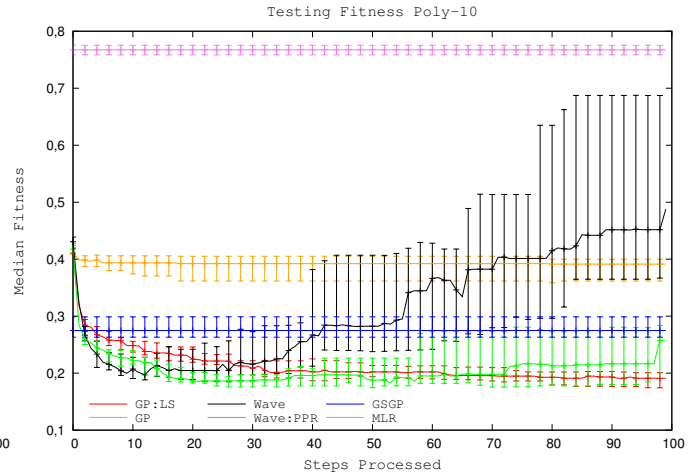
(a) Concrete Strength



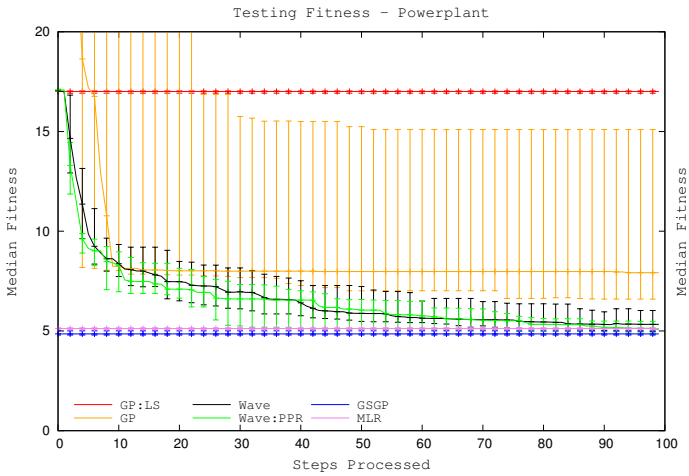
(b) Small Concrete



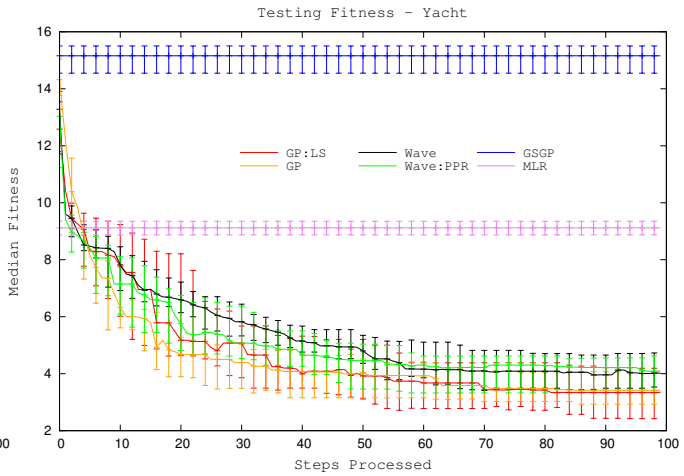
(c) Div-5



(d) Poly-10

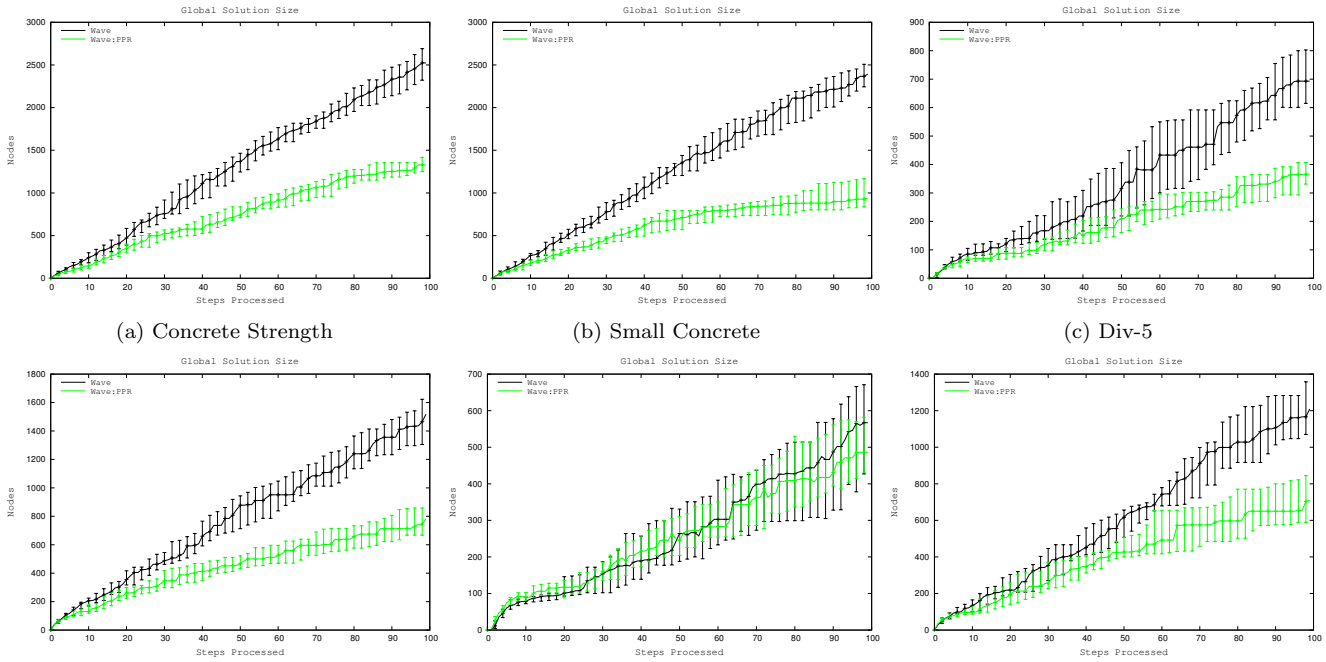


(e) Powerplant



(f) Yacht

Figure 2: Median Testing Fitness of joint solution / best individual - 95% confidence interval.



(a) Concrete Strength (b) Small Concrete (c) Div-5

(d) Poly-10 (e) Powerplant (f) Yacht

Figure 3: Median Size of Overall solution - 95% confidence interval.

[8] Stephen Jay Gould and Elisabeth S Vrba. Exaptation-a missing term in the science of form. *Paleobiology*, pages 4–15, 1982.

[9] Eva Jablonka, Marion J Lamb, and Anna Zeligowski. *Evolution in Four Dimensions, revised edition: Genetic, Epigenetic, Behavioral, and Symbolic Variation in the History of Life*. MIT press, 2014.

[10] Maarten Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In *Genetic programming*, pages 70–82. Springer, 2003.

[11] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[12] Barbara McClintock. *The significance of responses of the genome to challenge*. Singapore: World Scientific Pub. Co, 1993.

[13] David Medernach, Jeannie Fitzgerald, R. Muhammad Atif Azad, and Conor Ryan. Wave: A genetic programming approach to divide and conquer. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1435–1436, New York, NY, USA, 2015. ACM.

[14] David Medernach, Jeannie Fitzgerald, R. Muhammad Atif Azad, and Conor Ryan. Wave: Incremental erosion of residual error. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 1285–1292. ACM, 2015.

[15] Brook G Milligan. Punctuated evolution induced by ecological change. *American Naturalist*, pages 522–532, 1986.

[16] Alberto Moraglio, Krzysztof Krawiec, and Colin G Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII*, pages 21–31. Springer, 2012.

[17] R. Muhammad Atif Azad, David Medernach, and Conor Ryan. Efficient approaches to interleaved sampling of training data for symbolic regression. In *Nature and Biologically Inspired Computing (NaBIC), 2014 Sixth World Congress on*, pages 176–183. IEEE, 2014.

[18] Thi Hien Nguyen, Xuan Hoai Nguyen, Bob McKay, and Quang Uy Nguyen. Where should we stop? an investigation on early stopping for gp learning. In *Simulated Evolution and Learning*, pages 391–399. Springer, 2012.

[19] Luiz OVB Oliveira, Fernando EB Otero, Gisele L Pappa, and Julio Albinati. Sequential symbolic regression with genetic programming. 2014.

[20] Nir Oren. *Improving the effectiveness of information retrieval with genetic programming*. PhD thesis, 2002.

[21] Ludo Pagie and Paulien Hogeweg. Evolutionary consequences of coevolving targets. *Evolutionary computation*, 5(4):401–418, 1997.

[22] Riccardo Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In *Genetic programming*, pages 204–217. Springer, 2003.

[23] Ekaterina J Vladislavleva, Guido F Smits, and Dick Den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Evolutionary Computation, IEEE Transactions on*, 13(2):333–349, 2009.