A Comparison of Representations in Grammar-Guided Genetic Programming in the context of Glucose Prediction in People with Diabetes

Leon Ingelse
University of Lisbon
Jose Ignacio Hidalgo
Complutense University of Madrid
José Manuel Colmenar
King Juan Carlos University
Nuno Lourenço
University of Coimbra
Alcides Fonseca (me@alcidesfonseca.com)
University of Lisbon

Research Article

Keywords: Grammar-Guided Genetic Programming, Individual representations, Symbolic Regression, Initialization methods

Posted Date: November 21st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3596625/v1

Additional Declarations: No competing interests reported.

A Comparison of Representations in Grammar-Guided Genetic Programming in the context of Glucose Prediction in People with Diabetes

Leon Ingelse[1*], J. Ignacio Hidalgo[2], J. Manuel Colmenar[3], Nuno Lourenço[4], Alcides Fonseca[1*]

[1*]LASIGE, Faculdade de Ciências da Universidade de Lisboa, Lisboa, Portugal.
[2]Universidad Complutense de Madrid, Madrid, Spain.
[3]Universidad Rey Juan Carlos, Madrid, Spain.
[4]Universidade de Coimbra, Coimbra, Portugal.

*Corresponding author(s). E-mail(s): lingelse@lasige.di.fc.ul.pt; alcides@ciencias.ulisboa.pt;
Contributing authors: hidalgo@ucm.es; josemanuel.colmenar@urjc.es; naml@dei.uc.pt;

Abstract

The representation of individuals in Genetic Programming (GP) has a large impact on the evolutionary process. In this work, we investigate the evolutionary process of three Grammar-Guided GP (GGGP) methods, Context-Free Grammars GP (CFG-GP), Grammatical Evolution (GE) and Structured Grammatical Evolution (SGE), in the context of the complex, real-world problem of predicting the glucose level of people with diabetes two hours ahead of time. Our analysis differs from previous analyses by (1) comparing all three methods on a complex benchmark, (2) implementing the methods in the same framework, allowing a fairer comparison, and (3) analyzing the evolutionary process outside of performance. We conclude that representation choice is more impactful with a higher maximum depth, and that CFG-GP better explores the search space for deeper trees, achieving better results. Furthermore, we find that CFG-GP relies more on feature construction, whereas GE and SGE rely more on feature selection. Additionally, we altered the GGGP methods in two ways: using ϵ-lexicase selection, which solved the overfitting problem of CFG-GP; and with a penalization of complex trees, to create more interpretable trees. Combining ϵ-lexicase selection with CFG-GP performed best. Finally, we evaluated the impact of initialization methods in the

quality of solutions, having found that they have no impact, even when the change of representation has.

Keywords: Grammar-Guided Genetic Programming, Individual representations, Symbolic Regression, Initialization methods

1 Introduction

Genetic Programming (GP) can benefit from grammars to constrain the search space, giving rise to the area known as Grammar-Guided GP (GGGP) [1]. GGGP has been presented in many flavors, differing on the individual representation. The original GGGP algorithm used a Context-Free Grammar (CFG) to guide tree creation, and was thus named CFG-GP [2]. Shortly after, Ryan et al. [3] introduced the now-popular Grammatical Evolution (GE), which uses a linear representation [4]. GE's main benefits over tree-based GGGP are simplicity in implementation, speed of evolutionary operation execution, and its scarce memory consumption. GE also comes with multiple disadvantages, such as low locality and invalid individuals, resulting in performance deterioration and execution slow downs [5]. To remediate some of these issues, Lourenço et al. [6] developed Structured GE (SGE). SGE aims to reduce the locality disconnect seen in GE, which we elaborate on in section 2.2. As all GGGP methods use CFGs, we refer to CFG-GP as tree-based GGGP throughout this document to avoid confusion.

Since these methods have been proposed, they have been the focus of different comparisons [7–10]. Both tree-based GGGP and SGE showed better performance than GE, whereas superiority of either tree-based GGGP or SGE is disputed.

Problem Statement

Existing comparative studies [7–10] each have at least two of the following limitations:

- The comparison is done using toy benchmarks, not representative of real-world complexity from programs without a known, simple solution, and very noisy [7, 8, 10].
- Implementations of the different methods were not comparable, as they were implemented in different languages or frameworks. Therefore, performance differences cannot be attributed to the representation choice alone [7, 9].
- The evaluation is focused only on performance, without a characterization of the shape of programs throughout the evolution [7–10].

In this work we mitigate all three limitations. As such, the goal of this work is to provide an in-depth analysis of the performance of GE, SGE and tree-based GGGP, in which comparison is fair and in the context of a very complex, real-world regression problem. Furthermore, we aim to investigate the development of tree depth and feature occurrence throughout the evolution.

Approach

First, to address the limitations of the complexity of benchmarks, we selected the real-world, complex and noisy regression problem of glucose prediction (presented in

```
<exp>       ::= <exp> <op> <exp>
              | <variable>
              | <constant>
<op>        ::= + | / | *
<variable>  ::= x1 | x2 | x3
<constant>  ::= 1 | 2 | 3 | 4 | 5
```

Listing 1 Grammar example for Symbolic Regression.

section 4). This problem has no known solution, unlike previous studies, and it is very noisy where the testing error is not highly correlated with the training error (unlike Boston Housing [11], for example).

Second, differences may occur due to the methods implementations. For example, Lourenço et al. [9] did not constrain the maximum depth of GE, but contrained for SGE. Maximum depth limitation gives bias to smaller solutions, and therefore it substantially changes the search space. In our work, we use the same maximum depths for all three representations. Furthermore, the overhead of tree creating is the same in all three representations.

Third, to understand the effects of the representation of the individual in the evolutionary process in more detail, we selected a single benchmark and, besides fitness and error in testing, we also analyse the depth of generated trees throughout evolution, as well as the occurrence of relevant features. This way, we expose evolution characteristics of the representations.

Besides mitigating the above three limitations, we explored the impact of complexity-restraining methods on the performance of representation choice. To be precise, we explored the effect of reducing the maximum depth, introduction of ϵ-lexicase selection [12], and complexity penalties.

Futhermore, we also explore the impact of four initialization methods, Grow, PI-Grow, Full and Ramped Half and Half, in the evolution of each of these representations.

2 Grammar-Guided GP methods

In this section we introduce CFGs (Section 2.1) and the three GGGP representations (Section 2.2).

2.1 Context-Free Grammars

CFGs describe a language by a starting symbol, a set of terminal symbols, a set of non-terminal symbols and a list of productions. Listing 1 presents a BNF grammar for symbolic regression. The left-hand side (LHS) shows non-terminals, the right-hand side (RHS) shows non-terminals like `<exp>` and terminals like `+`. Each non-terminal in the LHS is linked to its *production rules* on the RHS. Individuals are created by expanding the starting symbol into a sequence of terminals and non-terminals, and recursively expanding non-terminals until none are left. The choices of productions taken during expansion characterizes an individual.

2.2 Grammar-Guided GP

Tree-based GGGP

Whigham et al. [2] introduced tree-based GGGP to support more expressive search-space definitions, via a CFG. Individuals are generated through the expansion of the grammar, using its production rules. Compared to standard GP, tree-based GGGP differs on the mutation and recombination operator, which are restricted by the CFG. The selection of compatible nodes in trees is traditionally considered more computationally expensive ($O(\log n)$) than selecting an element in a list ($O(1)$). [1] Additionally, the selection of a target node that was expanded from a non-terminal symbol is equiprobable. As such, and especially in recursive grammars, it is more common that the selected node is not a terminal. Therefore, individuals can be quite deep [10], which may result in a higher memory consumption, and bloat in regression problems [13].

Grammatical Evolution

Grammatical Evolution (GE) [3] represents individuals by linear strings, and uses a *genotype-to-phenotype* mapping to convert the representation (genotype) into a program (phenotype). GE has practical advantages over tree-based GGGP. Saving the linear representation is more memory-efficient than saving the whole tree. Additionally, implementing mutation and recombination operators in linear strings is easier and more efficient. A particular downside of GE is that changes in the linear representation do not produce proportional changes in the phenotype [5]. Because each element in the linear representation encodes a choice of productions to use in each expansion, changing from a terminal symbol to a non-terminal symbol will cause all the following values in the linear representation to be used in another context, to make other choices. This phenomenon is called *low locality* [7].

Structured GE

Structured GE (SGE) was introduced to counter the low locality that GE suffers from. Instead of having a single linear string, individuals in SGE are represented by a list of linear strings, where each linear string represents the production choices for a single non-terminal. Mutation and recombination operators are able to keep the same general order of expansion for each non-terminal symbol. However, SGE also has some shortcomings. Firstly, the locality is still lower than in tree-based representations as a small change in value might change a terminal expansion to be non-terminal, changing the expression of the remaining genotype. Because SGE requires pre-processing the grammar to understand the possible expansion of recursive non-terminal symbols [6], Lourenço et al. [14] proposed a dynamic approach that adjusts the length of the linear strings dynamically. The dynamic approach constrains the representation to an allowed maximum depth, similarly to tree-based representations. In this paper we consider this dynamic variant.

[1] However, the framework used for this evaluation, `GeneticEngine`, caches this information at each level so this overhead is not incurred time-wise, but rather in memory, which is not a problem in this particular application.

3 Related work

Before the introduction of SGE, O'Neill and Ryan [15] compared GE and tree-based GGGP, and found GE populations to be more diverse on the representation side, as distinct genotypes represented the same phenotype. However, Rothlauf and Oetzel [7] showed that this high-redundancy also leads to ineffective mutations and crossovers. Together with the low locality of GE, they argued that the fitness progression of GE resembles that of a random search algorithm. In fact, the low locality of the crossover in GE had been studied before [16, 17].

Integer and binary representations of GE have also been compared [18] with no major differences, thus we assume our work can also be extended to other linear representations.

Whigham et al. [8] studied the performance of GE and tree-based GGGP, and concluded that tree-based showed superior performance. For more information on the differences between tree-based GGGP and GE, see the discussion of Ingelse et al. [5].

When compared to both tree-based GGGP and GE on a set of benchmarks, SGE showed competitive performance to tree-based GGGP, and significant improvements to GE [10, 14]. Of note, individuals in tree-based GGGP were reported to be deeper than those in SGE, which is consistent with a previous report that GE had a structural bias towards smaller trees [19]. Furthermore, SGE was compared to GE on the same real-world problem as we do in this paper, and it significantly performed better than GE [9]. Notice that Lourenço et al. [9] excluded tree-based GGGP, that GE and SGE were not implemented in the same framework, and that only fitness progression was compared.

No study has compared all three GGGP methods on a real-world, noisy benchmark yet. Furthermore, all research was done across different frameworks, neglecting implementation details specific to each framework. Lastly, the above research fails to discuss the internal functioning of the algorithms, focusing on performance differences. In this paper, we aim to fill these gaps, and extend on the impact of different complexity-reduction methods on the performance of each representation.

4 Methodology

In this section we define the problem (Section 4.1) and elaborate on the experimental setup (Section 4.2).

4.1 Problem definition

Although usually benchmarks problems are used for the kind of investigations performed in this paper, we selected a complex real-world problem. The motivation is two-fold, first we have been working on it for several years and we are aware of the difficulty of the problem. Second, real-world problems have unpredictable search spaces that are difficult to explore. In particular, we tested the algorithms on a glucose prediction dataset formed by real data from patients suffering from diabetes.

People with diabetes suffer from a defect either in the secretion (type 1 diabetes) or in the action of insulin (type 2 diabetes)[2]. Insulin is a hormone secreted by the pancreas,

[2]There are other types of diabetes, but those are the two main types

which allows the processing of food so that the glucose it contains is metabolized and transferred to the cells of the body. When the insulin does not work or is not available, an increase of the levels of glucose in the blood is generated. If those high levels are maintained in time, it can lead to complications for the person with diabetes, not only in the short term but also in the long term. In the total absence of insulin secretion, the person has to administer doses of artificial insulin that make the effect of natural insulin in a person without diabetes and maintain glucose levels in an acceptable range for healthiness (usually between 70 and 180 mg/dl, for a person with diabetes.)

Deciding upon the amount of this dose is not an easy task, since there are many factors that affect glucose values. An excessive amount of insulin may produce an hypoglycemia event, while an insufficient dose may maintain the levels of glucose in dangerous levels. Hence, insulin dependant people need to somehow predict the value of glucose under a set of suppositions (insulin dosing and food to be ingested), and a set of recorder variables (such as historic values of glucose, ingested food, insulin dosing, exercise, etc.). Fortunately, nowadays, Continuous Glucose Monitoring Systems (CGM) can estimate interstitial glucose values every few minutes, and smartwatches and mobile applications can help to register important information and bio-metric measures. With this data, and the use of Genetic Programming. it is possible to develop prediction models to assist people with diabetes, as has been shown in [20] and [21].

As exposed, predicting or estimating the glucose value after a meal intake and/or insulin administration is a major problem for people with diabetes. The high variability, both within and among individuals, makes the problem very challenging. More formally, the problem can be defined as follows:

Find an expression for the value of the glucose in n minutes from the time of prediction ($\hat{G}_{t+n}$) as accurate as possible to the actual value, knowing information of the previous k minutes and supposing a set of actions in terms of food ingestion and insulin dosing.

In the past, several solutions for different time horizons, ranging from 15 to 240 minutes. In this paper we selected a 120 minutes time horizon. Two hours is the usual time taken by the person to re-evaluate if an additional insulin or meal action is needed, and is difficult enough to test our algorithms.

We tackled the problem as a symbolic regression problem which aims to find the expression for the predicted glucose in two hours, $\hat{G}_{t+120}$, as follows:

$$\hat{G}_{t+120} = f((G_{t+i}, i \in (-240 \dots 0)), (I_{t+i}, C_{t+i}, i \in (-240 \dots +120))$$

where G is the time series of blood glucose concentration values, I is the time series of insulin inputs, and C is the number of carbohydrate inputs as estimated by the patients. We consider that at each time point, t, the glucose data for the previous four hours is available, $G_{t+i}, i \in (-240 \dots 0)$, as well as insulin and carbohydrates data ranging from up to four hours before to (an estimation of) the coming two hours, $I_{t+i}, C_{t+i}, i \in (-240 \dots +120)$.

Table 1: Descriptive statistics for the data sets for all ten patients. Values in range 180-250 are (100% − Time glucose$_{<70}$ − Time glucose$_{>250}$ − Time in target$_{70-180}$).

Patient	Average (mg/dl)	Std. Deviation (mg/dl)	Time glucose <70 mg/dl	Time glucose >250 mg/dl	Time in range [70-180] mg/dl	# Points
1	157.67	62.25	4.48%	7.31%	37.57%	4018
2	145.44	64.02	8.33%	6.18%	41.83%	23534
3	143.46	45.45	2.14%	2.26%	49.18%	23821
4	150.47	56.70	4.12%	5.14%	41.29%	20090
5	139.17	67.93	14.17%	7.10%	41.89%	8036
6	142.58	60.08	10.08%	4.44%	41.21%	12628
7	176.33	68.35	3.65%	14.28%	27.07%	6888
8	135.34	46.11	4.85%	1.84%	54.06%	6027
9	146.82	59.82	7.54%	5.20%	39.49%	5740
10	166.13	86.12	9.33%	16.16%	35.69%	4305

Furthermore, in previous works [9, 20], 4 hours of information was shown to be enough to achieve good models and that more data could introduce noise to the data. Moreover, the usual separation among food intakes is around 4 hours. We use the same data and the same preprocessing used by Hidalgo et al. [20] and Lourenço et al. [9]. In those previous works, different variations of GP and GE were successfully used for generating symbolic regression models for prediction.

Table 1 shows patient details. Ten patients (n=10) were selected based on conditions of good glucose control for an observational study. Data from patients was acquired over multiple weeks using Medtronic continuous glucose monitors recorded at five-minute intervals. Each entry records the amount of carbohydrate intake as estimated by the patients, the amount of insulin injected, and the blood glucose value for a time instant.

As can be observed in Table 1 glucose data present high differences in average glucose, ranging from 135.34 ± 46.11 mg/dl of patient 8 to 176.33 ± 68.35 mg/dl of patient 7. The objective of the control of glucose is to maintain the values in range (or in target), i.e. in the [70-180] interval, as long as possible. Patients also present low values of the time in range (and high variability) which also indicates the difficulty of the prediction problem.

Data was divided for a k-fold cross-validation procedure. We chose k equal to 10 if enough data was available per fold (>10000 data points), or equal to 8 if not. Patients 1, 8, 9, and 10 were split into 8 folds, whereas the others into 10 folds.

4.2 Experimental setup

For the implementation of the experiments, we used GeneticEngine [22], a GGGP framework in Python that supports the three GGGP approaches presented in Section 2. GeneticEngine uses the same tree-generation method for each algorithm, diminishing performance differences due to implementation. We considered the root mean squared error between the prediction and the actual glucose values as the fitness function.

When possible, we used the same parameters for each GGGP method to ensure a fair comparison. Furthermore, we used the same grammar and parameter values from Lourenço et al. [9]. In Table 2 we detailed the parameter values. Notice that

Table 2: Parameters used for each GGGP algorithm, following Lourenço et al. [9].

	Value		
Parameter	**Tree-based**	**GE**	**SGE**
Number of runs	30		
Population size	200		
Generations	250		
Selection	Tournament with size 3		
Elitism	10%		
Crossover rate	90%		
Crossover	Tree-based	Point-wise	Structured
Mutation rate	100%		
Mutation	Tree mutation	Point-wise with $p = 0.05$	
Initial tree depth	Range from 4 - 8		
Maximum tree depth	19		

the maximum depth is not the same as the previous work. This is because we used the maximum tree depth, whereas Lourenço et al. [9] report the maximum depth of the expansion of recursive non-terminals, allowing for non-recursive symbols to be expanded after that limit is reached. For the sake of understanding the depth of trees, we chose to report the maximum tree depth (2 levels deeper than the maximum recursion for this grammar). Furthermore, we used the same tree initialization method for all GGGP approaches: PI Grow [23] until the minimum depth is reached, and then Random production. Lastly, we used the same grammar as used by Lourenço et al. [9].

The configuration specified in Table 2 depicts the standard configuration. To better understand the GGGP methods, we designed three alternative scenarios. The first alternative scenario uses a maximum depth of 10 (instead of 19), to understand the impact of limited maximum tree depth. We also experimented with intermediate depths, but we chose to report depth 10 as, on average, best individuals of GE and SGE reached depth 9 in the specified number of generations. Hence, using a maximum depth of 10 does not give an advantage to any of the methods.

The second alternative scenario uses ϵ-lexicase selection [12] instead of tournament selection. ϵ-lexicase selection is a regression extension of standard lexicase selection [24], which can perform multi-objective optimization for symbolic regression [25]. Where in tournament selection an individual of the population has a high fitness when it performs well on only one objective, ϵ-lexicase selection drives individuals to perform well on multiple objectives. Given multiple objectives, each individual is evaluated on the objectives in order, eliminating individuals for selection each time an objective is evaluated, and the individual's performance is below the ϵ threshold. Each time that an individual is selected for reproduction, the objectives are reordered, so that other objectives are given a higher importance. We used the median absolute deviation as the ϵ value, as recommended[12]. To use ϵ-lexicase selection for our regression problem, we split our datasets up into subsets and defined the performance on each subset as an objective. We split the datasets into the same number of subsets as there are individuals in our population (200), as we found this to work well.

The third and last alternative scenario incorporates a penalization of trees with a higher number of nodes, a measure of the complexity of the individual. We penalized 20% of the fitness of each individual, by multiplying it by the number of nodes divided by 100. As such, more nodes results in a worse fitness. We divide by 100 to reduce the effect of the penalty. The 20% penalty and normalizing value of 100 were chosen arbitrarily but without being insignificant and without ignoring the original fitness value.

5 Results

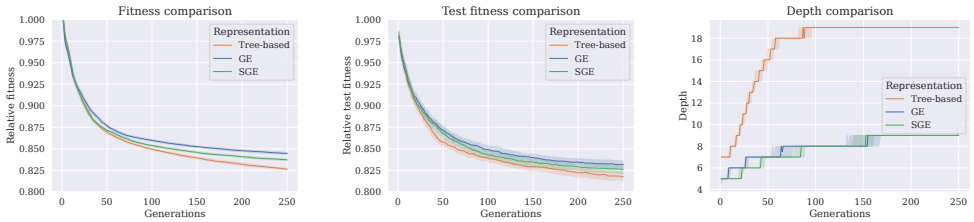
In this section, we discuss the evaluation methods we used (Section 5.1). After that, we analyze the evolutionary process of the GGGP algorithms (Section 5.2). Then, we study the impact of the two complexity restricting methods (Section 5.3). Finally, we evaluate the impact of different initialization methods (Section 5.4).

5.1 Analysis Methodology

We compared the performance of the GGGP methods in two ways. First, we analyzed the fitness in the last generation for each patient individually. We ran the problem for each fold of each patient, using the respective training and testing data. To show statistical significance, we trained and tested the model with 30 different seeds. In Section 5.2 we compare the results of the train and test results respectively for each patient. We performed a pairwise statistical analysis with the best average using Welch's two-sided t-test [26], as done by Uriot et al. [27]. As our comparisons are pairwise, our conclusions are also pairwise. We depict different background colors for each approach-patient combination, based on the p-value: bright green ($p > 0.1$, including the best approach), green ($0.1 \geq p > 0.05$), blue ($0.5 \geq p > 0.01$) and dark blue ($0.01 > p$). Furthermore, per alternative scenario, we underlined the best GGGP method(s) if it had a significant difference with at least one other GGGP method ($p > 0.1$).

Secondly, we analyzed the fitness progression throughout the evolutionary process. In order to visualize the evolutionary process for each representations for all runs, we normalized the best fitness in each generation to the best fitness of GE in the first generation of training. Hence, we created figures visualizing fitness improvement compared to the best fitness in the first generation of GE. Figure 1 is an example of this plot. Figure 1 shows all the runs for each fold of each patients normalized over the best fitness of GE in the first generation, for each of the three representation methods. The solid line represents the median of the runs, and the light shade the 95% bar. Note that the relative fitness is always the first generation of training of GE, regardless of the representation we are considering, or whether we are looking for training or testing data.

Additionally, we also report the depth with the median and the 95% error bar and the proportion of feature occurrence to all nodes for the best individuals throughout the evolutionary process (Figure 2). We depict feature occurrence with the average and the standard deviation bar.



(a) Relative fitness in training. (b) Relative fitness in testing. (c) Tree depth.

Fig. 1: Comparison of the training fitness, testing fitness, and tree depth of the best individuals for the GGGP methods with maximum depth 19. Fitness values are normalized to the best fitness of GE in the first generation.

5.2 Evolutionary process

Table 4: Comparison of training fitness in different approaches and methods for GGGP.

P	Standard			Depth 10		
	Tree-based	GE	SGE	Tree-based	GE	SGE
1	47.5 (±1.5)	48.5 (±1.6)	48.1 (±1.4)	48.1 (±1.4)	48.3 (±1.4)	47.9 (±1.3)
2	47.9 (±2.9)	50.3 (±2.7)	49.3 (±2.2)	48.8 (±2.2)	50.0 (±2.6)	49.0 (±2.2)
3	36.3 (±1.0)	36.8 (±1.0)	36.7 (±1.2)	36.5 (±0.8)	36.7 (±0.8)	36.7 (±1.3)
4	47.0 (±1.0)	47.5 (±0.9)	47.2 (±0.8)	47.2 (±0.8)	47.4 (±0.8)	47.1 (±0.9)
5	55.9 (±1.2)	56.8 (±1.1)	56.3 (±1.1)	56.3 (±0.8)	56.6 (±1.1)	56.2 (±1.1)
6	49.5 (±1.5)	50.8 (±1.5)	49.9 (±1.2)	50.1 (±1.1)	50.6 (±1.4)	49.8 (±1.1)
7	59.8 (±1.8)	61.6 (±1.5)	61.3 (±1.3)	61.0 (±1.5)	61.5 (±1.5)	61.1 (±1.5)
8	41.8 (±1.5)	42.9 (±1.2)	42.4 (±1.2)	42.7 (±1.3)	42.5 (±1.2)	42.3 (±1.2)
9	47.3 (±1.4)	48.3 (±1.2)	47.8 (±1.1)	48.0 (±1.2)	48.3 (±1.4)	47.8 (±1.2)
10	70.1 (±2.9)	72.7 (±2.7)	72.2 (±2.4)	71.8 (±2.6)	72.2 (±2.7)	72.0 (±2.3)
P	**ϵ-lexicase**			**Complexity penalties**		
	Tree-based	GE	SGE	Tree-based	GE	SGE
1	47.8 (±1.3)	48.4 (±1.2)	48.4 (±1.2)	49.7 (±1.6)	51.1 (±3.2)	50.8 (±3.3)
2	45.6 (±1.9)	47.4 (±1.8)	47.5 (±1.8)	52.6 (±2.4)	52.5 (±2.7)	52.5 (±3.2)
3	35.6 (±0.5)	36.2 (±0.6)	36.1 (±0.6)	37.9 (±1.2)	40.0 (±3.0)	39.5 (±2.7)
4	46.6 (±0.6)	47.2 (±0.5)	47.2 (±0.5)	48.6 (±1.3)	50.9 (±3.6)	50.9 (±3.6)
5	56.1 (±0.7)	56.6 (±0.7)	56.6 (±0.6)	57.8 (±1.2)	59.3 (±3.4)	58.0 (±2.1)
6	48.8 (±1.0)	49.8 (±1.0)	49.8 (±1.1)	52.5 (±1.3)	53.6 (±2.5)	52.2 (±2.1)
7	60.0 (±1.3)	61.3 (±1.2)	61.5 (±1.3)	63.7 (±1.5)	64.0 (±1.6)	63.5 (±1.6)
8	41.4 (±1.4)	42.2 (±1.7)	42.7 (±1.3)	44.4 (±0.8)	44.3 (±1.5)	44.2 (±1.2)
9	47.6 (±1.0)	48.4 (±1.0)	48.3 (±1.0)	49.8 (±1.2)	52.9 (±4.0)	50.6 (±2.8)
10	69.6 (±2.1)	72.1 (±2.2)	72.4 (±2.0)	75.6 (±2.8)	76.4 (±3.6)	75.6 (±3.2)

The first observation is that the evolution using tree-based GGGP achieves better fitness after generation 50 until the threshold of 250 (Figure 1a), which is backed by a statistical difference for all patients in this standard scenario (in the first column block in Section 5.2). The same improvement can be seen when using the same fitness function on the test dataset, albeit with a higher standard deviation (Figure 1b). However, we

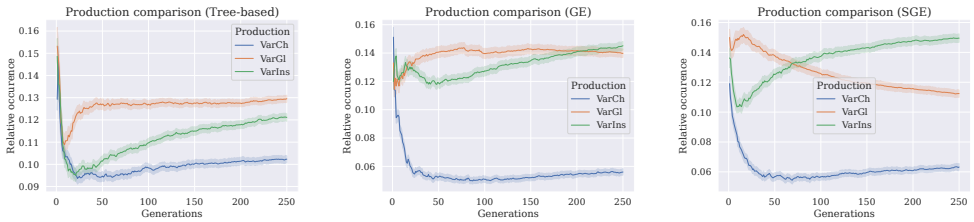
Table 5: Test fitness comparison.

P	Standard			Depth 10		
	Tree-based	GE	SGE	Tree-based	GE	SGE
1	50.3 (±12.3)	49.1 (±8.0)	50.1 (±19.3)	49.8 (±16.6)	49.1 (±8.2)	48.9 (±8.3)
2	47.8 (±5.5)	50.2 (±5.8)	49.2 (±5.6)	48.7 (±5.7)	49.9 (±6.2)	48.8 (±5.5)
3	63.5 (±468.6)	36.8 (±3.6)	36.8 (±3.8)	36.5 (±3.8)	36.8 (±3.9)	36.7 (±3.9)
4	47.1 (±3.7)	47.5 (±3.4)	47.3 (±3.4)	47.3 (±3.2)	47.5 (±3.3)	47.2 (±3.4)
5	56.4 (±5.0)	56.9 (±4.9)	56.5 (±4.8)	56.5 (±4.9)	56.8 (±4.8)	56.3 (±4.8)
6	49.9 (±4.4)	51.0 (±4.3)	50.2 (±4.3)	50.3 (±4.3)	50.8 (±4.3)	50.0 (±4.2)
7	123.7 (±917.1)	63.5 (±16.8)	63.0 (±12.5)	62.6 (±11.3)	63.9 (±20.4)	62.7 (±11.1)
8	42.5 (±5.2)	43.4 (±5.3)	42.8 (±5.3)	43.3 (±5.1)	42.9 (±5.4)	42.5 (±5.2)
9	49.6 (±9.6)	49.1 (±5.1)	48.8 (±6.8)	49.2 (±6.2)	49.1 (±5.1)	48.6 (±5.4)
10	71.6 (±10.4)	73.7 (±11.6)	73.3 (±11.7)	72.5 (±10.9)	73.0 (±11.5)	72.9 (±11.5)
P	**ϵ-lexicase**			**Complexity penalties**		
	Tree-based	GE	SGE	Tree-based	GE	SGE
1	48.9 (±7.9)	48.8 (±7.8)	48.8 (±7.6)	49.9 (±8.5)	51.3 (±8.9)	50.7 (±8.8)
2	45.5 (±5.1)	47.4 (±5.3)	47.4 (±5.5)	52.7 (±6.3)	52.2 (±6.4)	51.8 (±5.8)
3	35.8 (±3.3)	36.3 (±3.5)	36.1 (±3.5)	37.9 (±4.2)	39.9 (±5.0)	39.7 (±4.6)
4	46.7 (±3.1)	47.3 (±3.1)	47.2 (±3.1)	48.5 (±3.6)	50.9 (±5.0)	50.5 (±4.7)
5	56.6 (±4.8)	56.8 (±4.7)	56.8 (±4.6)	57.7 (±5.1)	59.5 (±6.6)	58.8 (±6.2)
6	49.3 (±4.5)	49.9 (±4.2)	50.0 (±4.3)	52.5 (±4.5)	53.6 (±5.0)	52.5 (±5.1)
7	62.2 (±14.0)	62.3 (±10.7)	62.7 (±11.3)	64.3 (±10.4)	64.7 (±10.4)	64.1 (±10.2)
8	42.0 (±5.9)	42.6 (±5.5)	42.9 (±5.3)	44.9 (±5.2)	44.6 (±5.2)	44.2 (±5.2)
9	48.7 (±5.7)	49.0 (±5.0)	49.0 (±4.4)	50.2 (±5.0)	54.2 (±7.2)	51.8 (±6.6)
10	70.8 (±10.1)	72.5 (±11.0)	72.9 (±10.9)	75.7 (±12.4)	77.2 (±14.2)	76.3 (±13.3)

notice that for patients 3 and 7, the average test fitness is exceptionally worse. Further analysis shows that for certain folds and runs, tree-based GGGP models overfit the training data, resulting in much worse test results. We attribute this overfitting to the complexity of the evolved trees. Notice that we have not observed overfitting with smaller trees. The last observation is that, despite the initial maximum depth and overall maximum depth being the same for all representation methods, tree-based consistently obtains deeper trees for its best individuals (Figure 1c). Right in the first generation, tree-based has deeper trees, and immediately starts to obtain deeper and deeper trees (until the maximum depth of 19), with the growth in GE and SGE is quite limited (only up to a depth of 8 and 7, respectively).

From these results we can confirm the results obtained in previous work. Like in Lourenço et al. [9], SGE performs better than GE, and like in the evaluation of Tree-based vs GE in Whigham et al. [8], we also conclude that tree-based has better performance in this context. Additionally, like in Lourenço et al. [10], we see that a tree-based representation obtains deeper trees than SGE. The presence of outliers in the testing dataset only in tree-based shows that obtaining deeper trees can result in overfitting (such outliers were not present in training). However, because they appeared only in a few runs (7 of 2760 runs), we do not consider this problematic, as these cases can be excluded by running three separate runs.

The growing depth in tree-based GGGP can also be attributed to the effect of the crossover operator. The crossover of tree-based GGGP happens on trees. When selecting two tree nodes, the algorithm might choose a deep node on the first tree,



(a) Tree-based GGGP (b) GE (c) SGE

Fig. 2: Dataset features occurrences throughout the evolution in the best individuals for the GGGP methods with depth 19.

and a shallow node with deep child nodes on the second tree. When these nodes and their corresponding subtrees are crossed over, the offspring will be deeper than its parents. On the other hand, for GE, crossover only increases the depth of its offspring when the end of the linear string representation contributes to this. In SGE, crossover only switches production choices, and, as in our example only a single non-terminal contributes to depth (¡*exp*¿), it can thus not create deeper offspring. We attribute the performance superiority of tree-based GGGP to its capability to exploit the depth better. In Section 5.3 we research this further.

Lastly, we compared the ratio of feature occurrences proportionate to the total number of nodes in the best individuals for tree-based GGGP, GE and SGE in Figure 2. In this case, *VarCh*, *VarGl* and *VarIns* are the three features corresponding to the three input variables for carbohydrates, glucose and insulin, respectively. As seen in the figure, the feature occurrences throughout the evolution show similarities. The proportion of each feature is the same at initialisation, and then diverges for the different features. The carbohydrate occurrences diminish, and insulin and glucose occurrences increase. As seen, the best individuals focus on glucose features firstly but, throughout the evolution, insulin features become more prominent in the three algorithms.

We make three observations. First, we notice that insulin and glucose are more frequent in the best individuals. Therefore, we argue that this information is more important in our models than information on carbohydrates for glucose prediction: An interpretation for this could be that glucose data already included carbohydrates effects. We also noted that carbohydrates data is sparse on the dataset, as a lot of entries are null.

Second, when comparing GE and SGE, SGE selects insulin features quicker. The reason of this velocity difference can be accounted for by the SGE-specific mutation and crossover operations, which do not influence other production choices as much as they do in GE. This exposes the superiority of SGE over GE.

Finally, when comparing tree-based GGGP to GE and SGE, we see that the variation in relative occurrence among features is three times lower (0.10 - 0.13 in tree-based GGGP to 0.06 - 0.15 in GE and SGE). From this we conclude that GE and SGE rely more on feature selection than tree-based GGGP does. Supposing that insuline and glucose values hold the most important information for future glucose prediction,

(a) Relative fitness in training. (b) Relative fitness in testing. (c) Depth.

Fig. 3: Comparison of the training fitness, testing fitness, and tree depth of the best individuals for the GGGP methods with maximum depth 10. Fitness values are normalized to the best fitness of GE in the first generation.

and as tree-based GGGP performs best in training, tree-based GGGP compensates the need of feature selection with the structure of the tree. As tree-based GGGP creates deeper trees (Figure 1c), we expect tree-based GGGP creating redundant subtrees with irrelevant features, also known as bloat [13]. Therefore, GE and SGE phenotypes will be more interpretable, as the trees will only hold relevant features.

In order to test these observations, we decided to run our experiments with different complexity-restricting methods.

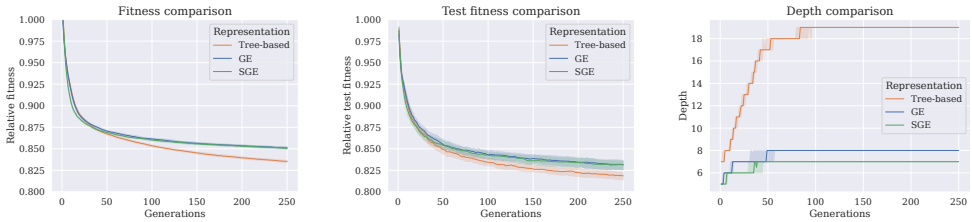
5.3 Complexity-restricting methods

Because the difference in performance between tree-based and SGE originated from being able to reach different depths, we evaluated the same scenario, but with a maximum depth of 10, instead of 19. Figure 3 presents the results for this scenario. We can see a similar evolution, but with smaller differences in fitness and depth. SGE reaches a better fitness on average, but without statistical differences per patient, as seen in Section 5.2.

We conclude that the SGE is less affected by the maximum depth than tree-based GGGP. Tree-based GGGP performs better for higher maximum depths as it exploits the depth better. With a smaller limit, both approaches behave similarly.

The second alternative scenario used the ϵ-lexicase selection [12] method. Figure 4 depicts a similar behaviour compared to the standard scenario with tournament selection, with tree-based achieving the allowed maximum depth, and with GE and SGE limited to half of the depth. The performance of tree-based in ϵ-lexicase is also better compared to each of the other methods in training (Section 5.2), tied or surpassed only by tree-based on the standard scenario. In testing it consistently performs better or ties with each other method (Section 5.2).

ϵ-lexicase selection is useful for this problem when combined with tree-based GGGP with a depth of 19, as it performs best for all patients. Most strikingly, ϵ-lexicase selection solved the issue of overfitting for tree-based GGGP, even though the depth exploitation still occurs. Moreover, the difference in performance between GE and SGE is reduced by ϵ-lexicase selection.



(a) Relative fitness in training. (b) Relative fitness in testing. (c) Depth.

Fig. 4: Comparison of the training fitness, testing fitness, and tree depth of the best individuals for the GGGP methods with ϵ-lexicase selection. Fitness values are normalized to the best fitness of GE in the first generation.

(a) Relative fitness in training. (b) Relative fitness in testing. (c) Depth.

Fig. 5: Comparison of the training fitness, testing fitness, and tree depth of the best individuals for the GGGP methods with complexity penalties. Fitness values are normalized to the best fitness of GE in the first generation.

In the third alternative scenario, we penalized the fitness proportional to the number of nodes in the tree, as presented in Section 4.2. In Figure 5, we see that no approach was able to achieve deep trees, and train performance of tree-based and SGE was similar. Still, tree-based generalized better than SGE, as it performed better in testing (Section 5.2).

5.4 Initialization Methods

We now explore the usage of four different initialization methods to understand their impact in all three representations. We consider the traditional Grow method, Position Independent Grow (PIGrow) [28], Full and Ramped Half and Half. We used a maximum depth of 14 in these experiments.

Figure 6 presents the prediction errors of the different initialization methods across the first four patients (the remaining are omitted as they reflect the same pattern). It is clear that the tree-based representation has a lower error across all initialization methods. In training data, tree-based representations evolve to achieve lower error, regardless of the initialization procedure. In fact, tree-based representation is

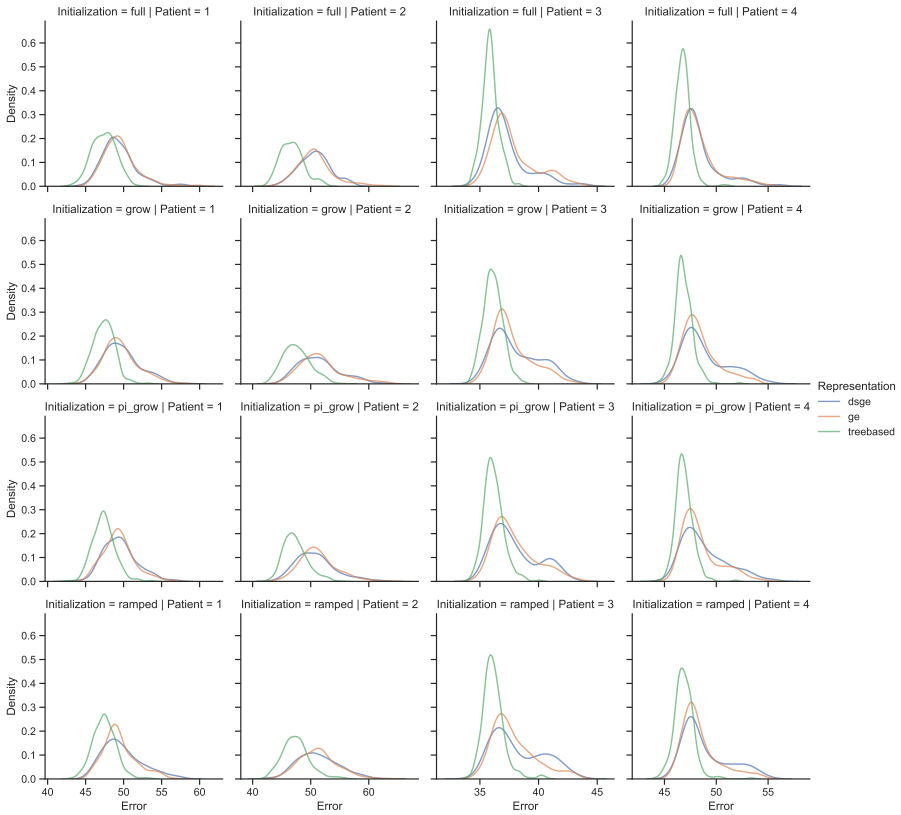


Fig. 6: Training data prediction error of all methods in different initialization methods, across the first four patients.

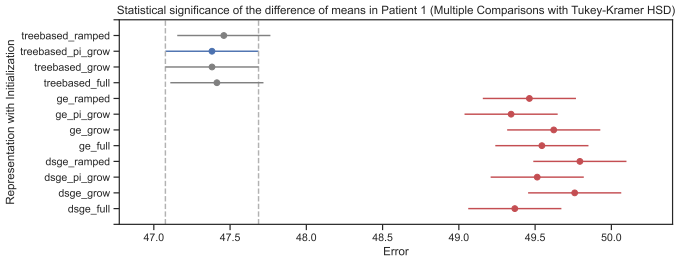


Fig. 7: Statistical comparison of the mean of different pairs of representation and initialization methods in patient one. Blue identifies the lowest mean. Red identifies the statistical different conditions. Grey conditions are not statistically different.

statistically better than any of the other approaches, but when varying the initialization

method, there is no statistical significance (Figure 7 evidences this difference in Patient 1, and the same occurs for all other patients).

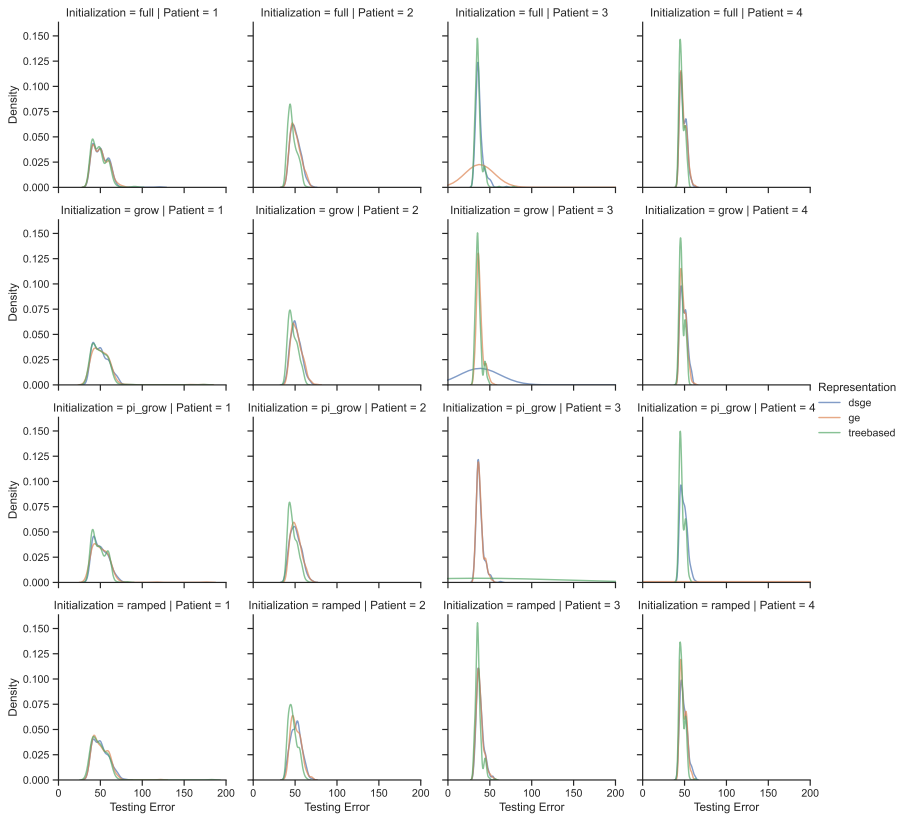


Fig. 8: Testing data prediction error of all methods in different initialization methods, across the first four patients.

In testing data, the results are much different, as seen in Figure 8. There is no statistical difference between any of the combinations of representation and initialization. The main reason, as identified before, is that the problem is too complex for the differences in training to be reflected in the testing data. Despite this fact, we can conclude that the initialization method plays no major role in this regression problem.

Furthermore, we also investigate some other metrics, related to the regression. In Figure 9, we confirm that tree-based is always the representation that achieves the deepest trees (resulting in better fitness in the training data). However, the differences between initialization methods are not statistically significant (Figure 10), showing that the initialization method does not play an important role in this problem. As for the number of nodes (Figure 11), the variance is high (especially in tree-based), so

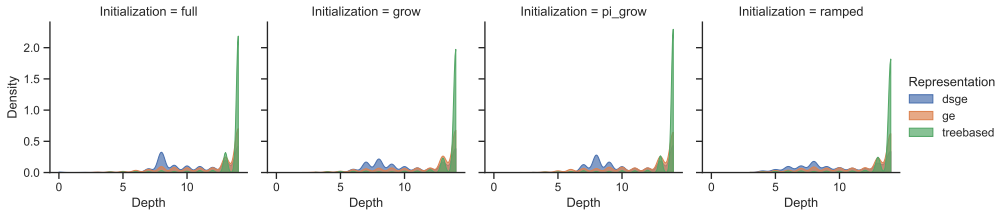


Fig. 9: Distribution of the depth across all initialization methods, across all patient data.

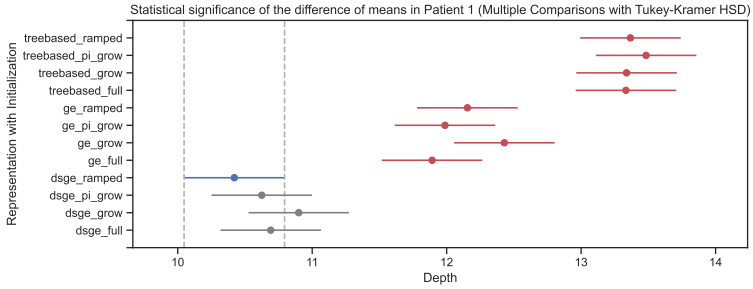


Fig. 10: Statistical comparison of the mean of the obtained depths of different combinations of representation and initialization methods for patient one. Blue identifies the lowest mean. Red identifies the statistical different conditions. Grey conditions are not statistically different.

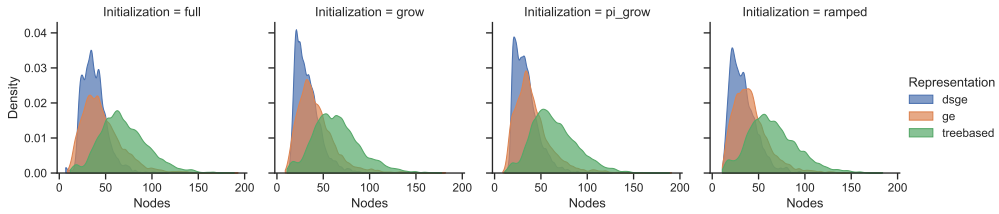


Fig. 11: Distribution of the number of nodes across initialization methods, for all patient data.

the means are not statistically significant, and there is also no major difference in the initialization method.

6 Conclusions

We investigated the evolutionary process of three GGGP approaches during the search for prediction models of blood glucose in people with diabetes. We concluded that

there are differences in the performance of GE, SGE and tree-based GGGP, even when every other parameter is the same. Tree-based GGGP reached deeper trees, with good performance, with the ability to perform better in training data. In the pairwise comparison, SGE and tree-based GGGP performed better than GE. Moreover, tree-based GGGP performed better than SGE in testing for some patients, and performed on par for other patients. This is in line with previous research [8–10], concluding those results hold when implemented on a single framework. However, we did not see superior performance when reducing the maximum depth to 10. Furthermore, tree-based GGGP overfitted the training data in a small number of cases, which was not seen in the other methods.

Apart from performance analysis, we researched differences within the evolutionary process. We found that, for this problem, GE and SGE relied more on the selection of features, whereas tree-based GGGP relied more on feature construction within the tree structure. As such, the individuals obtained by GE and SGE were more easily interpretable.

Next, we experimented with ϵ-lexicase selection and penalization of complexity in trees. Remarkably, ϵ-lexicase selection solved the overfitting issue for tree-based GGGP in this problem, while still exploiting the maximum tree depth. This combination, ϵ-lexicase selection with tree-based GGGP, gave the best performance. The complexity penalization we used saw performance deteriorate. However, trees were much smaller, and thus the method should be considered by practitioners if less-complex trees are desired, albeit in customized configurations.

Finally, we investigated the role of initialization methods in this problem, and we reported that there were no statistically differences between the initialization methods. We conclude that in symbolic regression problems with this pattern of complex data, the initialization method does not play a major role, when compared to the representation used.

A lesson for practitioners is to consider whether deep trees are wanted (better performance, possible lower interpretability). If they are desired, a tree-based representation should be used. However, overfitting can be a challenge in that scenario, and ϵ-lexi-case selection is a suggestion to avoid it. Finally, the downside of tree-based GP having a more tedious implementation is not applicable when using a framework that supports either representation with a parameter switch, such as offered by GeneticEngine.

Declarations

Ethical Statements

Non applicable.

Author Contributions

LI, JIH, HMC and AF designed the study. NL implemented a first version of the study. LI and AF implemented the final version of the study, on the current platform. LI and AF collected results and designed the visualizations. LI, JIH, JMC and AF wrote the paper. NL reviewed the paper.

Acknowledgements

We would like to thank the other authors of Genetic Engine, Guilherme Espada and Paulo Canelas.

Funding

This work was supported by *Fundação para a Ciência e Tecnologia* (FCT) in the LASIGE Research Unit under the ref. UIDB/00408/2020 and UIDP/00408/2020 and the CMU–Portugal project CAMELOT (LISBOA-01-0247-FEDER-045915), and the RAP project under the reference (EXPL/CCI-COM/1306/2021). JIH acknowledges the Spanish Ministerio de Innovación Ciencia y Universidad - grants PID2021-125549OB-I00 and PDC2022-133429-I00.

Data availability

Available under request to absys@ucm.es..

References

[1] Whigham, P.A.: Search bias, language bias, and genetic programming. Genetic Programming **1996**, 230–237 (1996)

[2] Whigham, P.A., *et al.*: Grammatically-based genetic programming. In: Proceedings of the Workshop on Genetic Programming: from Theory to Real-world Applications, vol. 16, pp. 33–41 (1995). Citeseer

[3] Ryan, C., Collins, J.J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) Genetic Programming, First European Workshop, EuroGP'98, Paris, France, April 14-15, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1391, pp. 83–96. Springer, ??? (1998). https://doi.org/10.1007/BFb0055930 . https://doi.org/10.1007/BFb0055930

[4] Freeman, J.J.: A linear representation for gp using context free grammars. In: Genetic Programming 1998: Proceedings of the Third Annual Conference, pp. 72–77 (1998). University of Wisconsin, Madison, Wisconsin USA. Morgan Kaufmann

[5] Ingelse, L., Espada, G., Fonseca, A.: Benchmarking individual representation in grammar-guided genetic programming. Technical report (2022)

[6] Lourenço, N., Pereira, F.B., Costa, E.: SGE: A structured representation for grammatical evolution. In: Bonnevay, S., Legrand, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) Artificial Evolution - 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers. Lecture Notes in Computer Science, vol. 9554, pp. 136–148. Springer, ??? (2015). https://doi.org/10.1007/978-3-319-31471-6_11 . https://doi.org/10.1007/978-3-319-31471-6_11

[7] Rothlauf, F., Oetzel, M.: On the locality of grammatical evolution. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S.M., Ekárt, A. (eds.) Genetic Programming, 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3905, pp. 320–330. Springer, ??? (2006). https://doi.org/10.1007/11729976_29 . https://doi.org/10.1007/11729976_29

[8] Whigham, P.A., Dick, G., MacLaurin, J., Owen, C.A.: Examining the "best of both worlds" of grammatical evolution. In: Silva, S., Esparcia-Alcázar, A.I. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, pp. 1111–1118. ACM, ??? (2015). https://doi.org/10.1145/2739480.2754784 . https://doi.org/10.1145/2739480.2754784

[9] Lourenço, N., Colmenar, J.M., Hidalgo, J.I., Garnica, O.: Structured grammatical evolution for glucose prediction in diabetic patients. In: Auger, A., Stützle, T. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019, pp. 1250–1257. ACM, ??? (2019). https://doi.org/10.1145/3321707.3321782 . https://doi.org/10.1145/3321707.3321782

[10] Lourenço, N., Ferrer, J., Pereira, F.B., Costa, E.: A comparative study of different grammar-based genetic programming approaches. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) Genetic Programming - 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10196, pp. 311–325 (2017). https://doi.org/10.1007/978-3-319-55696-3_20 . https://doi.org/10.1007/978-3-319-55696-3_20

[11] Asuncion, A., Newman, D.: UCI machine learning repository. Irvine, CA, USA (2007)

[12] Cava, W.G.L., Spector, L., Danai, K.: Epsilon-lexicase selection for regression. In: Friedrich, T., Neumann, F., Sutton, A.M. (eds.) Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016, pp. 741–748. ACM, ??? (2016). https://doi.org/10.1145/2908812.2908898 . https://doi.org/10.1145/2908812.2908898

[13] Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genet. Program. Evolvable Mach. **10**(2), 141–179 (2009) https://doi.org/10.1007/s10710-008-9075-9

[14] Lourenço, N., Assunção, F., Pereira, F.B., Costa, E., Machado, P.: Structured grammatical evolution: A dynamic approach. In: Ryan, C., O'Neill, M., Collins, J.J. (eds.) Handbook of Grammatical Evolution, pp. 137–161. Springer, ??? (2018). https://doi.org/10.1007/978-3-319-78717-6_6 . https://doi.org/10.1007/978-3-319-78717-6_6

[15] O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Trans. Evol. Comput. **5**(4), 349–358 (2001) https://doi.org/10.1109/4235.942529

[16] Keijzer, M., Ryan, C., O'Neill, M., Cattolico, M., Babovic, V.: Ripple crossover in genetic programming. In: Miller, J.F., Tomassini, M., Lanzi, P.L., Ryan, C., Tettamanzi, A., Langdon, W.B. (eds.) Genetic Programming, 4th European Conference, EuroGP 2001, Lake Como, Italy, April 18-20, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2038, pp. 74–86. Springer, ??? (2001). https://doi.org/10.1007/3-540-45355-5_7 . https://doi.org/10.1007/3-540-45355-5_7

[17] O'Neill, M., Ryan, C., Keijzer, M., Cattolico, M.: Crossover in grammatical evolution. Genet. Program. Evolvable Mach. **4**(1), 67–93 (2003) https://doi.org/10.1023/A:1021877127167

[18] Hugosson, J., Hemberg, E., Brabazon, A., O'Neill, M.: Genotype representations in grammatical evolution. Appl. Soft Comput. **10**(1), 36–43 (2010) https://doi.org/10.1016/j.asoc.2009.05.003

[19] Schweim, D., Thorhauer, A., Rothlauf, F.: On the non-uniform redundancy of representations for grammatical evolution: The influence of grammars. In: Ryan, C., O'Neill, M., Collins, J.J. (eds.) Handbook of Grammatical Evolution, pp. 55–78. Springer, ??? (2018). https://doi.org/10.1007/978-3-319-78717-6_3 . https://doi.org/10.1007/978-3-319-78717-6_3

[20] Hidalgo, J.I., Colmenar, J.M., Kronberger, G., Winkler, S.M., Garnica, O., Lanchares, J.: Data based prediction of blood glucose concentrations using evolutionary methods. J. Medical Syst. **41**(9), 142 (2017) https://doi.org/10.1007/s10916-017-0788-2

[21] Contador, S., Colmenar, J.M., Garnica, O., Velasco, J.M., Hidalgo, J.I.: Blood glucose prediction using multi-objective grammatical evolution: Analysis of the "agnostic" and "what-if" scenarios. Genetic Programming and Evolvable Machines, 1–32

[22] Espada, G., Ingelse, L., Canelas, P., Barbosa, P., Fonseca, A.: Data types as a more ergonomic frontend for grammar-guided genetic programming. In: Scholz, B., Kameyama, Y. (eds.) Proceedings of the 21st ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, GPCE 2022, Auckland, New Zealand, December 6-7, 2022, pp. 86–94. ACM, ??? (2022). https://doi.org/10.1145/3564719.3568697 . https://doi.org/10.1145/3564719.3568697

[23] O'Neill, M., Brabazon, A., Nicolau, M., Garraghy, S.M., Keenan, P.: π grammatical evolution. In: Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part II, pp. 617–629 (2004). Springer Berlin Heidelberg

[24] Spector, L.: Assessment of problem modality by differential performance of lexicase

selection in genetic programming: a preliminary report. In: Soule, T., Moore, J.H. (eds.) Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012, Companion Material Proceedings, pp. 401–408. ACM, ??? (2012). https://doi.org/10.1145/2330784.2330846 . https://doi.org/10.1145/2330784.2330846

[25] La Cava, W., Moore, J.H.: An analysis of ϵ-lexicase selection for large-scale many-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 185–186 (2018)

[26] Welch, B.L.: The generalization of 'student's'problem when several different population varlances are involved. Biometrika **34**(1-2), 28–35 (1947)

[27] Uriot, T., Virgolin, M., Alderliesten, T., Bosman, P.A.N.: On genetic programming representations and fitness functions for interpretable dimensionality reduction. In: Fieldsend, J.E., Wagner, M. (eds.) GECCO '22: Genetic and Evolutionary Computation Conference, Boston, Massachusetts, USA, July 9 - 13, 2022, pp. 458–466. ACM, ??? (2022). https://doi.org/10.1145/3512290.3528849 . https://doi.org/10.1145/3512290.3528849

[28] Fagan, D., Fenton, M., O'Neill, M.: Exploring position independent initialisation in grammatical evolution. In: IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016, pp. 5060–5067. IEEE, ??? (2016). https://doi.org/10.1109/CEC.2016.7748331 . https://doi.org/10.1109/CEC.2016.7748331