Contribution of Probabilistic Structured Grammatical Evolution to efficient exploration of the search space. A case study in glucose prediction

Jessica Mégane
jessicac@dei.uc.pt
University of Coimbra, CISUC/LASI, DEI
Coimbra, Portugal

Nuno Lourenço
naml@dei.uc.pt
University of Coimbra, CISUC/LASI, DEI
Coimbra, Portugal

J. Ignacio Hidalgo
hidalgo@ucm.es
Universidad Complutense de Madrid
Bioinspired Intelligence S.L.
Madrid, Spain

Penousal Machado
machado@dei.uc.pt
University of Coimbra, CISUC/LASI, DEI
Coimbra, Portugal

Abstract

People with Type 1 diabetes need to predict their blood glucose levels regularly to keep them within a safe range. Accurate predictions help prevent short-term issues like hypoglycemia and reduce the risk of long-term complications. Evolutionary algorithms have shown potential for this task by generating reliable models for glucose prediction.

This work compares four evolutionary approaches: Structured Grammatical Evolution (SGE), a float-based variant (SGEF), and two probabilistic methods, Probabilistic SGE (PSGE) and Co-evolutionary PSGE (Co-PSGE). These methods are tested on their ability to predict glucose levels two hours ahead in individuals with diabetes. Two aspects are examined: predictive performance and the diversity of the phenotypes produced by each approach.

Results indicate that SGEF provides statistically better performance than the other methods. Although PSGE and Co-PSGE do not show statistically significant improvements in prediction accuracy, they generate a broader set of solutions and explore more distinct areas of the search space.

CCS Concepts

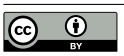
• **Computing methodologies** → **Genetic programming**.

Keywords

Grammar-based, Probabilistic, Structured Grammatical Evolution, Glucose Prediction

ACM Reference Format:

Jessica Mégane, Nuno Lourenço, J. Ignacio Hidalgo, and Penousal Machado. 2025. Contribution of Probabilistic Structured Grammatical Evolution to efficient exploration of the search space. A case study in glucose prediction. In *Genetic and Evolutionary Computation Conference (GECCO '25), July 14–18, 2025, Málaga, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3712256.3726444



GECCO '25, July 14–18, 2025, Málaga, Spain

ACM ISBN 979-8-4007-1465-8/2025/07
https://doi.org/10.1145/3712256.3726444

1 Introduction

Genetic Programming (GP) [22] is an Evolutionary Computation (EC) technique designed for the automatic evolution of programs. GP approaches have become popular due to their success in tasks such as program synthesis [14, 34, 41], symbolic regression modelling [3, 33, 42], neuroevolution [4, 8, 47], achieving results in several real-world applications such as glucose prediction in people with diabetes [15, 18, 25].

Over time, researchers have proposed distinct representations for solutions, variation operators, and other innovations. Among the most successful variants of GP are those that rely on grammars to decode solutions. Grammars have been used in GP to introduce syntactic constraints to the solutions generated [28], avoiding unnecessary search in unfeasible regions. The grammars can also be designed to explore specific areas of the search space or include domain knowledge about the problem at hand. The most well-known grammar-based GP approaches are Context-Free Grammar Genetic Programming (CFG-GP) [45] and Grammatical Evolution (GE) [36, 37].

Another successful subset of GP involves the use of probabilistic models to guide the search process [21, 40]. These methods often replace standard genetic operations, such as crossover and mutation, with resampling based on a learned probabilistic model. The combination of grammars with probabilistic models was introduced as early as the CFG-GP proposal [44]. Still, it was not until later that it gained more interest.

In probabilistic grammar-based approaches, the grammar encodes the syntax of the programs and assigns probabilities to each production rule, biasing the search, and also indicating the likelihood of generating a valid program. These probabilities are updated throughout the evolutionary process, often based on the best-fitted individuals [19, 30], following a specific distribution [46], or a predefined metric [7].

This study compares four approaches: standard Structured Grammatical Evolution (SGE) which uses an integer-based representation; SGE with Float representation (SGEF), which modifies its variation operators to match a float-based representation; and two probabilistic grammar-based variants, Probabilistic Structured Grammatical Evolution (PSGE) and Co-evolutionary Probabilistic Structured

Grammatical Evolution (Co-PSGE), both of which extend SGEF by evolving the probabilities of selecting each production rule of the grammar. In PSGE, the probabilities are updated during evolution based on production rules that create the fittest individual. In Co-PSGE, each individual maintains its own probability distribution over the grammar rules, which can change through mutations throughout the evolutionary process. To compare these methods, a set of challenging problems was selected, previously used to assess grammar-based GP approaches in the task of predicting glucose levels in people with diabetes [15]. The prediction model takes into account past glucose measurements, insulin doses, and carbohydrate intake to estimate glucose levels two hours ahead.

The experimental study includes two main analyses. First, the performance of each method is assessed using error-based metrics, supported by statistical testing. Second, the diversity of phenotypes among the best individuals is examined through a two-dimensional space generated using t-distributed Stochastic Neighbor Embedding (t-SNE) visualization [16].

The remainder of the paper is structured as follows: Section 2 presents the background on grammar-based probabilistic GP approaches. Section 3 introduces the four approaches used for the analysis. Section 4 details the problem and related work. Section 5 details the experimental setup and data used. Section 6 presents the experimental results. Finally, Section 7 summarizes the main findings and suggests directions for future research.

2 Background

Probabilistic grammar-based algorithms constitute a significant subset of GP methods [21, 28, 40], where probabilities are assigned to production rules, biasing the selection of rules during the generation of individuals. These approaches commonly use a Context-Free Grammar (CFG) to define the syntax rules as seen in CFG-GP [44], scalar and vector Stochastic Grammar-based GP (GP) [35], Grammar model-based program evolution (GMPE) [39], Context Free Grammar Transformation (CFGT) [6, 7], Univariate Model Building Grammatical Evolution (UMBGE) [19, 20], Probabilistic Linear GP (GP) [42], Probabilistic Grammatical Evolution (PGE) [30], PSGE [32], and Co-PSGE [31]. However, some extend these by incorporating context, using a Context Sensitive Grammar (CSG) as in Strongly Typed GP (GP) [43] and Grammar-based GP with Bayesian Network (GP) [46], or by leveraging Lindenmayer systems, as in the case of Program Evolution with Explicit Learning (PEEL) [38]. In addition to evolving the probabilities, GMPE [39], CFGT [6, 7], and PEEL [38] also evolve the structure of the grammar by dynamically introducing new rules.

Most of these methods, such as GMPE, SG-GP, PEEL, CFGT, and BGBGP fall under the category of Estimation Distribution Algorithms (EDAs). As such, they do not use standard genetic variation operators. Instead, they re-sample the population or a subset of individuals during the run, using a probabilistic model to guide the process. Other methods follow the principles of traditional evolutionary algorithms, applying variation operators such as crossover and mutation [30–32]. Hybrid approaches also exist, combining both strategies: part of the population is sampled from the learned probabilistic model, while the rest undergoes variation operators [44].

CFG		PCFG
NT	**Production rules**	**Probability Ranges**
<expr>	<expr><op><expr>	[0.00, 0.37]
	<var>	[0.37, 1.00]
<op>	+	[0.00, 0.22]
	-	[0.22, 0.39]
	*	[0.39, 0.68]
	/	[0.68, 1.00]
<var>	x	[0.00, 0.41]
	y	[0.41, 0.67]
	1.0	[0.67, 1.00]

Figure 1: Example of a CFG (left) and a PCFG (right). Both use the same set of production rules, but the PCFG includes probability ranges, where the size of each range corresponds to the rule's selection probability.

3 Structured Grammatical Evolution

SGE [26] is an alternative representation to GE, offering a different genotypic structure and mapping mechanism. In SGE, the genotype is a set of dynamic lists of ordered integers. Each list is associated with a non-terminal symbol of the CFG, and each element of the list represents the index of a derivation rule to be expanded to create the individual. An example of genotype is shown in Figure 2.

A CFG is a tuple $G = (NT, T, S, P)$ where NT and T represent the non-empty set of *Non-Terminal (NT)* and *Terminal (T)* symbols, S is an element of NT in which the derivation sequences start, called the axiom, and P is the set of production rules. The rules in P are in the form $A ::= \alpha$, with $A \in NT$ and $\alpha \in (NT \cup T)^*$. The NT and T sets are disjoint. Each grammar defines a language $L(G) = \{w : S \overset{*}{\Rightarrow} w, w \in T^*\}$, that is the set of all sequences of terminal symbols that can be derived from the axiom. An example of CFG is represented on the left of Figure 1.

The mapping from genotype to phenotype is performed by expanding production rules based on the codons in the genotype, always expanding the leftmost non-terminal. The selected rule is the one whose index appears first in the list associated with the non-terminal being expanded. If genotypic changes result in an empty list for a given non-terminal, new codons are randomly generated and appended as needed. To prevent excessive growth, a maximum tree depth is defined in advance. Once this limit is reached, only rules that lead to the shortest possible expansions are allowed. After selecting a derivation rule, the corresponding codon is added to the list for that non-terminal in the genotype.

Figure 2 shows an example of the genotype-phenotype mapping, using the grammar in Figure 1. The mapping starts with the axiom of the grammar, in this case <expr>, and is expanded to the production rule of index 0, since this is the first codon in the list associated with the non-terminal <expr>. The index corresponds to the production rule <expr><op><expr>, so the next token to expand is <expr>. The next codon in the list is 1 so <expr> will be expanded to the respective rule, i.e., <var>. Because this is the leftmost non-terminal, to expand <var> we have to take the first element of this non-terminal list, which is 2. Index 2 corresponds to the derivation rule 1.0. Since this corresponds to a terminal symbol, it becomes definitive, moving on to the next non-terminal, which is <op>. The process is repeated until there are no more non-terminals to expand.

Genotype

<expr>	<op>	<var>
[0, 1, 1]	[1]	[2, 0]

Derivation step	Codon used
<expr>	(0)
<expr><op><expr>	(1)
<var><op><expr>	(2)
1.0<op><expr>	(1)
1.0/<expr>	(1)
1.0/<var>	(0)
1.0/x	

Phenotype: `1.0/x`

Figure 2: Example of the genotype-phenotype mapping of SGE, using the grammar described in Figure 1.

Genotype

<expr>	<op>	<var>
[0.19, 0.46, 0.87]	[0.27]	[0.32, 0.64]

Derivation step	Codon used
<expr>	(0.19)
<expr><op><expr>	(0.46)
<var><op><expr>	(0.32)
x<op><expr>	(0.27)
x-<expr>	(0.87)
x-<var>	(0.64)
x-y	

Phenotype: `x-y`

Figure 3: Example of the genotype-phenotype mapping of a genotype represented as floats, with a PCFG.

The mutation operator changes a randomly selected codon, which represents a production rule, to another valid one. The crossover recombines two parents to generate the offspring. The offspring inherits the list of each non-terminal from one of the parents, and this decision is made based on a randomly generated binary mask. The mask contains a binary value for each list of the genotype (i.e., one for each non-terminal of the grammar) [26, 27].

3.1 SGE with Float Representation

While standard SGE uses an integer representation in the genotype, SGE variants [30–32] use floats as codons as an alternative representation of genotypes, which are then used to select the production rule of the grammar. In the remainder of this paper, we will refer to this representation as SGEF. The float representation requires a different mapping and mutation mechanism. The genotype mapping is done through a Probabilistic Context-Free Grammar (PCFG), maintaining the probabilities static throughout evolution. A PCFG is a quintuple $PG = (N, T, S, P, Probs)$, extending CFG with $Probs$, which is a set of probabilities associated with each production rule. Figure 1 shows an example of a PCFG.

The individuals are represented by a set of dynamic lists, each associated with a non-terminal of the grammar. The lists contain an ordered sequence of real numbers, bounded to the interval [0, 1], with each codon corresponding to the probability of choosing a production rule.

An example of an individual's mapping process is depicted in Figure 3, using the grammar in Figure 1. The process begins by expanding the axiom of the grammar, <expr>, using the first codon in the list of the respective non-terminal of the genotype, in this case 0.19. The non-terminal <expr> presents two derivation rules, with different probabilities of being chosen. The 0.19 codon is included in the range of probabilities of the first rule, <expr><op><expr>, therefore expansion is made for that rule. The derivation is always done from the leftmost non-terminal, so the next non-terminal to expand is <expr>. The next available codon in the list of the non-terminal <expr> is 0.46, which falls within the probability range of the second expansion rule, <var>. The <var> will be the next to expand and has three possible derivation rules. 0.32 is the first codon available in the list of <var> of the genotype, and falls within the range of probabilities covered by the first production rule, x, which is a terminal symbol. <op> is the next symbol to expand, using the first codon available in the respective list to select a production rule. In this case the codon is `0.27`, which corrresponds to the terminal symbol -. The procedure is repeated until a valid individual is formed.

The mutation operator randomly selects codons, based on the probability of mutation previously defined. A Gaussian mutation is applied to these codons, bounding the new values in the interval $[0, 1]$. These types of mutations have been widely used in the literature and have proven to be a good approach to make small changes in the search space [5, 9]. Figure 4 shows an example of Gaussian mutation in an individual. Assuming that the second codon of the non-terminal <expr> was randomly selected (0.46), and the value generated with a normal distribution of mean 0 and standard deviation 0.50 ($N(0, 0.50)$) was -0.17, the codon will now assume a value of 0.29 ($0.46 - 0.17 = 0.29$).

The crossover operator combines the genetic material of two individuals to generate an offspring, similarly to the crossover mechanism employed by SGE [26].

If after mutation and crossover, the mapping process generates an invalid phenotype, these can be randomly generated as needed. If the depth limit is surpassed, the algorithm considers only the rules that correspond to the shortest path, and adjusts the probabilities of each of them, so that the sum is 1. Using the new probabilities, the production rule is chosen with the normal procedure: It is verified whether the codon belongs to the probability range of each production rule of the non-terminal to be expanded and when this condition is verified, the rule is chosen.

Genotype before mutation:

<expr>	<op>	<var>
[0.19, **0.46**, 0.87]	[0.27]	[0.32, 0.64]

Gaussian mutation: `N(0, 0.50) = -0.17`

Genotype after mutation:

<expr>	<op>	<var>
[0.19, **0.29**, 0.87]	[0.27]	[0.32, 0.64]

Figure 4: Example of a mutation applied to codon 0.46 of the non-terminal <expr>. The mutation value was drawn from a Gaussian distribution with mean 0 and standard deviation, σ, 0.50, (N(0, 0.50)) was -0.17. After mutation, the resulting codon is 0.29 (0.46-0.17).

3.2 Probabilistic Structured Grammatical Evolution

PSGE [32] employs a float-based genotype representation, similar to SGEF, but differs by evolving the probabilities of the PCFG through generations. In PSGE, at the end of each generation, the probabilities of the PCFG are updated based on the frequency of the production rules used to map an individual. It alternates between the best individual overall and the best individual from the current generation. All genotypes are re-mapped according to the newly updated grammar, possibly forming new individuals.

The probabilities are updated by iterating thorugh each production i of each non-terminal j used during expansion of the current best individual or best individual overall. A *counter* tracks the number of times each production was chosen. The probability (*prob_i*) in the PCFG contains the current probability of selecting that production. If the count is greater than zero, that is, the production rule was used to map the individual, *prob_i* is updated using (1). If the count is zero, that is, the production rule has not been used by the individual, *prob_i* is updated using (2). The learning factor is represented by λ, with $\lambda \in]0, 1]$, and is used to make the transitions on the search space smoother.

$$prob_i = min(prob_i + \lambda * \frac{counter_i}{\sum_{k=1}^{j} counter_k}, 1.0) \quad (1)$$

$$prob_i = prob_i - \lambda * prob_i \quad (2)$$

After updating the probabilities using the equations, these are normalized, to ensure that the sum of the probabilities for the production rules of each non-terminal equals 1.

3.3 Co-evolutionary Probabilistic Structured Grammatical Evolution

Co-PSGE [31] extends SGEF by assigning to each individual its probability distribution associated with the rules of the grammar, used to map the genotype. At the end of each generation, the probabilities of the PCFG of each individual are itself subject to variation through mutations, with Gaussian mutations applied.

This process is demonstrated in Figure 5. There are two parameters for this process, the mutation rate (between 0 and 1) and the normal distribution standard deviation. The mutation probability is used to check for each non-terminal production rule whether mutation should occur. In is shown in Figure 5 the example of this process for the non-terminal <expr>. When a production is selected for mutation (step 1 of the mutation process depicted in Figure 5), it is generated a value from a Gaussian distribution of mean 0 and the standard deviation previously defined (step 2), in which in the example the value generated is 0.08. The generated value is added to the value of the production rule selected (step 3) keeping the new value in the range [0,1], and then the values of the other rules are adjusted so the sum of the probabilities of all the production rules of a non-terminal is 1. The grammar of the parent with the best fitness is passed on to the offspring during crossover. The individual's phenotype is updated at the end of each generation using the new grammar and genotype.

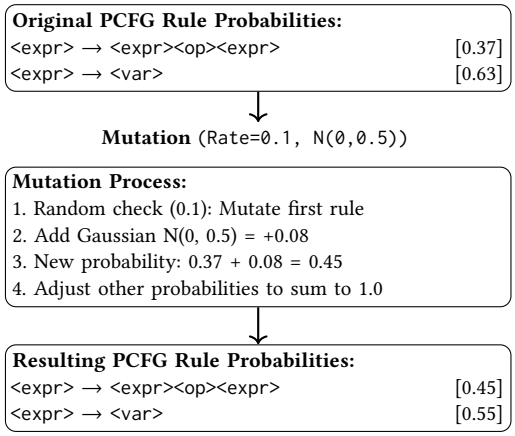


Figure 5: Example of PCFG probability mutation process employed in Co-PSGE. A production rule is randomly selected for mutation based on the mutation rate previously defined. The probability associated to the selected rule is modified by adding a value drawn from a Gaussian distribution N(0, σ).

4 Glucose Prediction

Insulin is a hormone, naturally produced by the pancreas, responsible for breaking down sugar molecules (glucose) and converting them into energy for the body. Diabetes mellitus is a metabolic disease where the body either does not produce enough insulin or does not respond to it properly, leading to abnormally high glucose levels in the blood.

The number of people living with the disease is continually increasing, with around 537 million adults suffering from the condition in 2021 [12], which equates to approximately 1 in 10 adults. Type 1 diabetes is an autoimmune disease in which the immune system destroys the cells of the pancreas, usually permanently, so that little or no insulin is produced [1]. Type 2 diabetes is diagnosed when the body develops resistance to insulin, although the pancreas might still be able to produce it [13]. This is the most common type of diabetes, constituting more than 90 per cent of the cases.

There are several ways to control glucose levels. Most Type 2 patients who produce insulin rely on medication, while Type 1 patients inject synthetic insulin. Despite the use of medication, diet and exercise play a very important role in managing blood sugar levels and preventing cardiovascular disease. Another important factor in controlling the disease is monitoring blood glucose levels. Continuous Glucose Monitoring (CGM) devices provide real-time information about blood sugar levels, which is not only less invasive than doing multiple finger pricks a day, but also gives patients and their healthcare providers more data to better understand how levels fluctuate with food intake, exercise or even stressful events. For patients who inject insulin, it facilitates the process of calculating the amount to inject, to keep blood glucose within a healthy range. Too much insulin can cause low blood sugar (hypoglycaemia), which over a prolonged period can lead to organ failure, permanent brain damage or even death [10]. High blood sugar (hyperglycaemia) can cause heart problems or even stroke [11].

The data collected by CGM devices allows different machine learning algorithms to develop predictive models to help people

with diabetes [2, 23, 24]. Among these approaches, grammar-based GP algorithms have also been studied in diabetes prediction problems and have shown promising results.

Hidalgo et al. [15] introduced a dataset comprising data from 10 patients at a public hospital in Spain. The dataset includes information collected over several weeks using Medtronic CGMs, recorded at 5-minute intervals, along with estimates of carbohydrate intake provided by the patients and the amount of insulin injected. This same dataset was used in a study [15], where two Evolutionary Algorithms (EAs), namely GP and GE, were compared with two machine learning algorithms, such as random forests and k-nearest neighbors. In this study, predictions were made considering forecasting windows of 30, 60, 90, and 120 minutes. The results showed that GE and GP presented better or similar performance when compared with the other approaches, with the solutions generated being smaller and potentially better suited for integration into wearable devices. Additionally, SGE [26] was compared with GE in the glucose prediction problem [15] using the same dataset and grammar, but only 120-minute windows were considered. Results were consistent with previous works, with SGE showing better results in both training and testing. The resulting models also proved to be more robust, with statistically significant differences observed for all patients.

Recently, a more in-depth analysis was performed [17] on performance, tree depth, and feature selection in grammar-based algorithms SGE, CFG-GP, and GE. All algorithms were compared using the same framework and parameters. Results showed that tree-based CFG-GP performed best on training data, resulting in deeper trees, but performed similarly or better than SGE on testing data. With a smaller tree depth limit, SGE had better average fitness, though without statistical differences. GE and SGE relied more on feature selection, making their solutions more interpretable, while CFG-GP relies more on feature construction. A follow-up study [18] analyzed different initialization methods but found no statistical differences, highlighting the stronger impact of representation.

5 Experimental Study

To understand how the different probabilistic approaches affect the search, we compare PSGE and Co-PSGE with standard SGE (with integer representation of solutions), and SGEF (with a float representation). This enables a fair comparison between the approaches, helping to understand if the differences observed are due to the representation or the evolution of the probabilities.

The dataset and grammar used to predict glucose values is the one proposed by Hidalgo et al. [15], and subsequently used by SGE [25] and other grammar-based GP algorithms [17]. We use data from 10 patients, which contains information on the amount of carbohydrate consumed (estimated by the patients), the amount of insulin injected, and the blood glucose value. The solutions are evaluated with the Root Mean Squared Error (RMSE) between the predicted values and the target values, with the goal of minimizing the error.

In this paper, we processed the data so that the part of the dataset used for training and the part used for testing had similar averages. Table 1 presents the average of the blood glucose (mg/dl) data on train and on test.

Table 1: Mean and standard deviation of the blood glucose (mg/dl) data on train and on test.

Patient	Train	Test
1	159.01 ± 49.27	158.77 ± 38.96
2	151.35 ± 51.47	151.16 ± 39.08
3	142.69 ± 36.78	143.02 ± 32.21
4	154.81 ± 47.66	237.58 ± 32.55
5	141.12 ± 55.91	140.94 ± 41.65
6	147.48 ± 55.58	147.65 ± 55.29
7	176.34 ± 65.13	176.46 ± 63.11
8	136.03 ± 44.43	136.31 ± 45.42
9	148.59 ± 56.50	148.21 ± 55.56
10	169.59 ± 83.24	169.40 ± 82.78

Table 2: Parameters used in the experimental analysis.

	Algorithm			
Parameter	**SGE**	**SGE Float**	**PSGE**	**Co-PSGE**
Number of runs	30			
Population size	200			
Generations	500			
Selection method	Tournament, size 3			
Elitism	10%			
Crossover rate	0.9			
Mutation rate	0.05			
Mutation	Int. flip	Gaussian N(0, 0.5)		
Max. Init. Depth	6			
Max. Tree Depth	17			
Learning Factor	-	-	0.01	-
PCFG Mutation Rate	-	-	-	0.025
PCFG Mutation	-	-	-	N(0, 0.5)

5.1 Parameters

The experiments were conducted using the parameters previously employed by grammar-based GP approaches in this problem [17, 25], which are common for all the four approaches. In particular, crossover and the selection operators such as tournament and elitism operate comparably across all four algorithms. Because of the differences in the genotype representation, the mutation is different between the approaches. On SGE the mutation changes the codon with a different integer, in PSGE, Co-PSGE, and SGEF the mutation changes the codon by adding a Gaussian mutation of distribution N(0, 0.5) (for more details, refer to Section 3.1). The probabilistic variants, PSGE and Co-PSGE present additional parameters related to the update of the probabilities of the grammar. Specifically, the learning factor in PSGE, while Co-PSGE incorporates the probability of mutating the probabilities of the grammar, and the values from the Gaussian distribution to generate the mutation. The probabilities assigned to the production of the grammars for PSGE and Co-PSGE, are uniform at the beginning of the evolutionary process. Table 2 shows the parameters for each algorithm. The code used in the experiments is publicly available[1].

6 Experimental Analysis

Two different analysis were conducted in this study. First, a performance analysis, followed by an analysis of the phenotypic space explored by the four algorithms.

6.1 Performance Results

To evaluate the performance of the algorithms, we average the results of 30 runs, with a Stratified 8-fold crossvalidation, followed by a statistical analysis for model comparison. Since the populations

[1] https://github.com/jessicamegane/

Table 3: Median and standard deviation of lowest RMSE obtained on the train dataset, and effect size (r). Values in bold show the algorithm that presented to be better with statistical differences when compared with SGE. Results are average of 30 runs.

	SGE	SGEF		PSGE		Co-PSGE	
P	Median	Median	r	Median	r	Median	r
1	47.92 ± 1.14	**47.41 ± 1.81**	+	47.89 ± 2.40	∼	48.16 ± 1.87	∼
2	48.48 ± 2.37	**47.72 ± 4.26**	+	49.06 ± 4.62	∼	49.98 ± 3.00	-
3	36.49 ± 1.21	**36.30 ± 1.65**	+	37.34 ± 2.11	-	36.60 ± 2.09	-
4	47.03 ± 1.01	**46.71 ± 1.12**	+	47.01 ± 1.88	∼	47.14 ± 1.50	∼
5	56.12 ± 0.82	**55.90 ± 1.36**	+	56.21 ± 2.30	-	56.24 ± 1.59	-
6	49.63 ± 1.22	**49.13 ± 2.16**	+	49.63 ± 2.75	∼	49.80 ± 1.64	-
7	61.23 ± 0.89	**60.18 ± 1.76**	++	61.00 ± 2.49	∼	61.35 ± 1.12	-
8	42.30 ± 1.06	**41.02 ± 1.45**	++	**41.57 ± 1.38**	+	42.72 ± 1.15	-
9	47.72 ± 1.05	**47.31 ± 1.85**	+	47.68 ± 2.00	∼	47.84 ±2.23	-
10	71.54 ± 2.11	**70.69 ± 3.69**	+	71.47 ± 3.03	∼	72.84 ± 3.32	-

Table 4: Median and standard deviation of lowest RMSE obtained on the test dataset, and effect size (r). Values in bold show the algorithm that presented to be better with statistical differences when compared with SGE. Results are average of 30 runs.

	SGE	SGEF		PSGE		Co-PSGE	
P	Median	Median	r	Median	r	Median	r
1	47.91 ± 1.82	**47.40 ± 2.32**	+	47.89 ± 2.40	∼	48.13 ± 2.32	∼
2	48.63 ± 2.47	**47.84 ± 4.34**	+	49.06 ± 4.62	∼	49.89 ± 3.01	-
3	36.31 ± 1.31	36.16 ± 1.76	∼	37.34 ± 2.11	-	36.53 ± 2.23	-
4	46.80 ± 1.19	**46.26 ± 1.26**	+	47.01 ± 1.88	∼	46.77 ± 1.64	∼
5	55.99 ± 1.25	55.77 ± 1.70	∼	56.13 ± 2.51	∼	56.16 ± 1.82	∼
6	49.74 ± 1.32	**49.44 ± 2.36**	+	49.63 ± 2.75	∼	49.97 ± 1.75	∼
7	60.78 ± 1.26	**60.08 ± 1.87**	+	61.00 ± 2.49	∼	61.07 ± 1.34	-
8	41.97 ± 1.27	**40.67 ± 1.66**	++	**41.57 ± 1.38**	+	42.56 ± 1.30	-
9	47.42 ± 1.21	**46.89 ± 1.98**	+	47.68 ± 2.00	∼	47.65 ± 2.40	-
10	71.83 ± 2.46	71.16 ± 5.13	∼	71.84 ± 3.47	∼	73.05 ± 3.73	-

were independently initialized and the results did not meet the criteria for parametric tests, the Mann-Whitney non-parametric test with Bonferroni correction was applied to assess meaningful differences between the methods. To determine the significance of the differences, the effect size, `r`, was calculated. In Tables 3 and 4 the "∼" sign was used when there were no significant differences between samples. When there are differences, "+" or − denotes that the approach is better or worse, respectively, with a small effect size (`r <= 0.3`). "++" or "−−" represents a medium effect size (`0.3 < r <= 0.5`), and "+++" or "−−−" indicates a large effect size (`r > 0.5`). For all the statistical tests we considered a significance level of $\alpha = 0.05$.

Table 3 and Table 4 show the median and standard deviation of the best individuals in train and test, respectively, of each run and fold, for each algorithm. Additionally, we show the effect size of the statistical tests which compare SGEF with SGE, PSGE with SGE, and Co-PSGE with SGE, to represent possible statistical differences between the approaches. In both tables, SGEF presents lower median values, for all patients. It presents statistical differences for all patients in training and for seven patients in testing, compared to SGE. These results indicate that different representations have significant impact on the results of glucose prediction, which is in line with the analysis presented by Ingelse et al. [17, 18].

PSGE shows no statistical differences compared to SGE for most patients, performing worse for Patient 3 and 5 but better for Patient 8, where statistical differences are present. PSGE only shows statistical differences compared to SGE for Patients 3 and 5, performing worse, and Patient 8 performing better. In testing, the results are similar except for Patient 5, where no statistical differences are observed. Co-PSGE is statistically worse than SGE for most patients. In training, it shows similar distributions for Patients 1 and 4. In testing, no statistical differences are observed for four patients (Patients 1, 4, 5, and 6).

By analysing the medians of the best individuals, we can generally identify which model performed best in training and testing for each patient. However, since this is a real-world problem, the performance over 30 runs is less relevant when considering patient treatment. The main objective of an algorithm in this context is to produce the best possible regression model for predicting glucose values. To assess this, we focus on the best individual overall for each algorithm. Table 5 presents the training and test values of the best individual overall for each algorithm and patient. Additionally, we report the lowest test error obtained. In the table, values of the "Best Individual" column in bold indicate the individual overall with the lowest test error for each patient. In the "Best Test" column, bold values highlight the algorithm that achieved the lowest test error, while underlined values indicate test errors lower than those of SGE.

By analysing the train and test values of the best individual in train for each algorithm ("Best Individual" column), we observe that SGE had the best individual performance in test for two patients (patients 4 and 6), SGEF for four patients (Patients 1, 2, 7, 8), Co-PSGE for three patients (Patients 5, 9, 10) and PSGE achieved the best only for Patient 3. This indicates that the alternative approaches generally outperform SGE, with PSGE being the least effective among them, as it underperformed the others on all patients but one. However, when compared directly to SGE, PSGE still yielded better individuals for half of the patients.

Looking closely at the values of the best individuals, the differences in error between most approaches are relatively small. However, for Patients 5 and 10, SGE shows much higher errors than the other approaches. These two patients generally exhibit higher errors across all approaches, suggesting that the algorithms have more difficulty to find a good model for them. Interestingly, Co-PSGE found the best solutions for these two patients, despite being statistically worse than SGE in training, and in test for Patient 10. Previous work [29] showed that PSGE and Co-PSGE generate more unique solutions than SGE in the benchmark analysed, indicating greater population diversity. While this diversity can help in discovering better solutions, it can also lead to worse, which may explain why Co-PSGE, despite its capacity of finding the best solutions, does not outperform SGE on average across all runs. This suggests that Co-PSGE might require more evolutionary time (generations) to surpass the other approaches.

The "Best Test" column of Table 5 of each algorithm represents the best test value obtained. These values are particularly important as they indicate which algorithm is more likely to provide the best model in real-world scenarios of prediction glucose values for patients with diabetes. Table 5 shows that SGEF is the one who is able to obtain the lowest test errors most frequently (values in bold), presenting the lowest value than the other approaches in half of the patients. To better analyse the performance of each algorithm compared to SGE, we underlined the test errors lower than those of SGE. PSGE, SGEF and Co-PSGE were able to obtain a lower test error than SGE for 8, 7 and 6 patients, respectively. Patient 4 is the

Table 5: Training and test errors of the best overall individual and the best test-set individual for each algorithm, shown per patient. Bold values indicate the algorithm with the lowest error. Underlined values mark results that are better than those obtained by SGE.

	SGE			**SGEF**			**PSGE**			**Co-PSGE**		
	Best Individual		Best Test	Best Individual		Best Test	Best Individual		Best Test	Best Individual		Best Test
P	Train	Test		Train	Test		Train	Test		Train	Test	
1	45.75	46.09	45.12	**44.74**	**44.51**	43.91	45.07	46.12	44.22	44.78	44.83	**43.79**
2	43.76	43.54	43.54	**43.61**	**43.24**	**43.24**	43.91	43.47	43.47	43.79	44.06	44.06
3	35.23	34.79	34.36	35.05	35.83	34.50	**35.13**	**34.09**	**34.09**	35.41	34.98	34.73
4	**45.24**	**44.04**	**44.04**	45.19	44.84	44.65	45.18	45.83	44.51	45.21	44.20	44.20
5	54.65	55.93	53.35	54.29	53.78	52.93	54.35	54.75	52.91	**54.52**	**52.83**	**52.83**
6	**47.54**	**47.15**	47.12	47.26	47.22	47.15	47.47	48.59	**46.99**	47.64	47.68	47.13
7	58.93	58.87	58.20	**57.52**	**57.93**	**55.89**	58.42	58.99	56.33	58.42	59.38	57.48
8	38.88	39.52	38.55	**37.96**	**37.79**	**37.29**	38.84	39.40	37.70	38.46	39.02	38.31
9	45.68	44.89	44.66	44.80	45.42	**42.66**	45.67	45.60	44.87	**44.99**	**44.36**	44.23
10	67.19	68.73	65.94	66.05	64.95	**64.29**	65.81	67.92	65.49	**66.51**	**64.84**	64.60

only where SGE got better test value than the other approaches. It is worth mentioning that for this patient, PSGE and Co-PSGE did not show statistical differences compared to SGE, indicating they were equivalent.

This comparison highlights that the alternative representation of SGE not only achieves good values but may also provide a more effective approach overall. Additionally, the evolution of the probabilities of PSGE and Co-PSGE enables them to find better solutions than SGE in most cases. While PSGE does not show statistical significance for most patients, it is the approach that was able to find the best test results for 8 patients, making it the most suitable choice for a real-world application. Additionally, Co-PSGE proves to be a viable choice as well, particularly in more complex scenarios, where it is able to find the best solutions for more challenging instances like Patients 5 and 10.

6.2 Phenotype analysis

The previous analyses have shown that the algorithms behave differently, and that in the case of algorithms with float representation, including probabilistic approaches, they generally seem to find better solutions. To see the differences between the algorithms, we analysed the solutions in terms of their phenotype. The first analysis consists of visualising the best individuals from each run and fold of each algorithm, projected in two dimensions using the t-SNE [16] method. The t-SNE embeddings were created using the best 15 individuals from each fold, run and algorithm, i.e. 14400 samples (8 folds, 30 runs, 15 individuals, 4 algorithms).

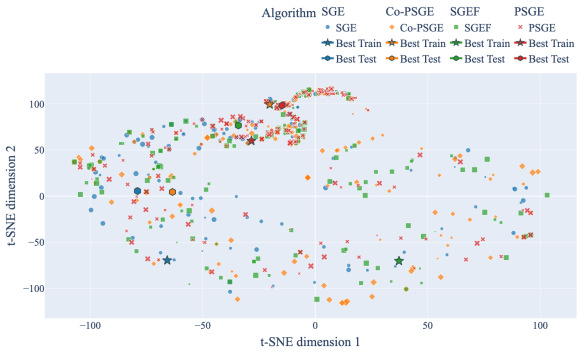
Figure 6 displays the t-SNE visualization for the four algorithms in different generations. Each algorithm's best individual from every run and fold is projected into a two-dimensional space, allowing for a clear comparison of their differences within the search space. SGE is represented as blue circles, Co-PSGE as orange diamonds, SGEF as green squares, and PSGE as red crosses. The size of the symbols is related to their error: smaller shapes present higher error, and bigger shapes, smaller errors. The best individual in training is marked with a star, while the best in testing is represented by a hexagon, each colored according to the corresponding algorithm.

Looking at the image from generation 500 (Figure 6d), it is noticeable that the algorithms explore mostly distinct regions of the search space. SGE predominantly covers the left side of the plot, and SGEF occupies mostly the right side. Co-PSGE and PSGE are more widely distributed. By looking at the evolution of the solution over generations, we observe that they are more concentrated on the top left at generation 50 (Figure 6a), and slowly expand to larger areas of the search space. However, looking at generation 500 (Figure 6d) only Co-PSGE and SGEF present solutions in the top right corner, suggesting that these algorithms explore more unique regions of the phenotypic space.

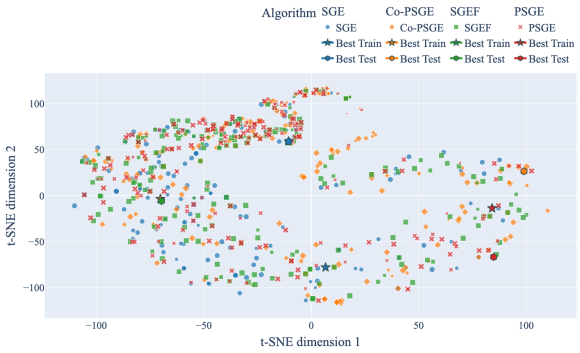
When comparing all algorithms, it becomes clear that the best solutions from the SGE variants (SGEF, PSGE, and Co-PSGE) are predominantly located on the right side of the plot. In contrast, SGE's best training solution is on the left, and its best test solution lies in the middle of the lower quadrants. Since SGEF, PSGE, and Co-PSGE consistently find better solutions than SGE, and the best solutions are concentrated on the right side, this could indicate that SGE's limited exploration of this region might be a key reason for its difficulty in finding the best solutions, despite having a good performance overall. Another intriguing aspect of this analysis is the proximity of certain solutions. For instance, SGEF's Best Test and Co-PSGE's Best Test are close to each other, and Co-PSGE's Best Train is near PSGE's Best Train and Test, with SGEF's Best Train also nearby. This raises the question of what these solutions might have in common. One observation in the SGE solutions is that they tend to be smaller. This aligns with findings from prior studies [18], which demonstrated that SGE generates shallower trees, explaining why its solutions are less complex. However, these differences may also be attributed to similar patterns in the phenotype or the features selected, which requires further investigation to analyze potential differences between the methods.

7 Conclusions and Future Work

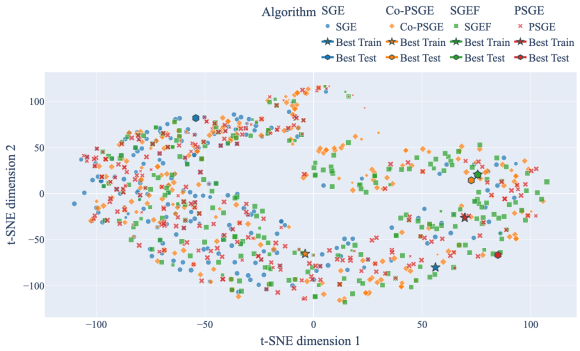
In this paper we compared four approaches for glucose level prediction in people with diabetes: standard SGE (which uses an integer representation); SGE with float representation (SGEF), which modifies variation operators to suit its representation; and two probabilistic grammar-based variants, PSGE and Co-PSGE, which extend SGEF by evolving grammars that assign probabilities to each production rule. We presented two different analysis. First, we evaluated the performance of each approach in terms of error,



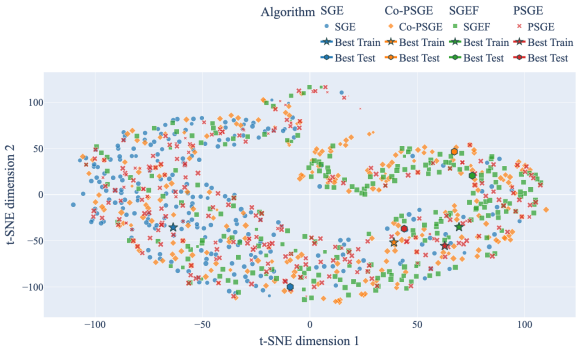
(a) t-SNE for the best individuals on generation 50, for Patient 1.



(b) t-SNE for the best individuals on generation 100, for Patient 1.



(c) t-SNE for the best individuals on generation 250, for Patient 1.



(d) t-SNE for the best individuals on generation 500, for Patient 1.

Figure 6: t-SNE visualization of the best individuals of each algorithm at generation 500 for Patient 1. SGE is represented as blue circles, Co-PSGE as orange diamonds, SGEF as green squares, and PSGE as red crosses. Best Train individuals are represented by a star, and the Best Test individuals are represented by an hexagon.

presenting the statistical tests. Second, we investigated the phenotypes distribution of the best individuals in a two-dimensional space created using t-SNE.

The results showed that the float based representation (SGEF) is better than standard SGE for most patients in testing. Additionally, we observed that all three SGE variants consistently found better individuals in test than SGE. Specifically, PSGE, SGEF and Co-PSGE surpassed SGE in 8, 7 and 6 patients out of 10, respectively.

In the second analysis, we saw that the projection of the phenotypes of each method is spread differently on the search space. PSGE, SGEF and Co-PSGE's best individual overall, in training and in testing, are close to each other, while SGE's solutions are in an opposite area of the search space. A previous work showed that Co-PSGE presented more unique solutions than PSGE and standard SGE [29], and that is also observed in this paper. Co-PSGE was the approach which more uniformly covered the search space. The results indicated that the probabilistic approaches might need more generations to surpass SGE statistically, despite being able to find better solutions.

Future work should involve a more detailed analysis of the phenotypes, including the size and distribution of selected production rules. In the case of probabilistic approaches, analyzing the probabilities could complement the interpretation of the results. Additionally, it would be valuable to compare the distribution of solutions with the performance results for each patient, as this paper only provides a brief analysis for Patient 1.

Acknowledgments

This work was supported by Project No. 7059 - Neuraspace - AI fights Space Debris, reference C644877546-00000020, supported by the RRP - Recovery and Resilience Plan and the European Next Generation EU Funds, by Portuguese Recovery and Resilience Plan (PRR) through project C645008882-00000055, Center for Responsible AI, and by national funds by FCT - Fundação para a Ciência e a Tecnologia, I.P., in the framework of the Project UIDB/00326/2025 and UIDP/00326/2025. The first author is funded by FCT under the grant 2022.10174.BD. J.I.H acknowledges support by Spanish Government AEI grants PID2021-125549OB-I00 and PDC2022-133429-I0 grant funding by MCIN/AEI /10.13039/501100011033 and European Union Next Generation EU/ PRTR.

References

[1] Ammira Al-Shabeeb Akil, Esraa Yassin, Aljazi Al-Maraghi, Elbay Aliyev, Khulod Al-Malki, and Khalid A. Fakhro. 2021. Diagnosis and treatment of type 1 diabetes at the dawn of the personalized medicine era. *Journal of Translational Medicine* 19, 1 (2021), 137. doi:10.1186/s12967-021-02778-6

[2] Md Ashraful Alam, Amir Sohel, Kh Maksudul Hasan, and Mohammad Ariful Islam. 2024. Machine Learning And Artificial Intelligence in Diabetes Prediction And Management: A Comprehensive Review of Models. *Journal of Next-Gen Engineering Systems* 1, 01 (Dec. 2024), 107–124. doi:10.70937/jnes.v1i01.41

[3] Muhammad Sarmad Ali, Meghana Kshirsagar, Enrique Naredo, and Conor Ryan. 2021. Towards Automatic Grammatical Evolution for Real-world Symbolic Regression. (2021), 68–78. doi:10.5220/0010691500003063

[4] Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. 2018. DENSER: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines* 20, 1 (Sept. 2018), 5–35. doi:10.1007/s10710-018-9339-y

[5] H. Beyer and H. Schwefel. 2004. Evolution strategies – A comprehensive introduction. *Natural Computing* 1 (2004), 3–52.

[6] Peter AN Bosman and Edwin D de Jong. 2004. Grammar transformations in an EDA for genetic programming. (2004).

[7] Peter AN Bosman and Edwin D de Jong. 2004. Learning Probabilistic Tree Grammars for Genetic Programming. In *International Conference on Parallel Problem Solving from Nature*. Springer, 192–201.

[8] Henrique Branquinho, Nuno Lourenço, and Ernesto Costa. 2023. SPENSER: Towards a NeuroEvolutionary Approach for Convolutional Spiking Neural Networks. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. ACM, New York, NY, USA, 2115–2122. doi:10.1145/3583133.3596399

[9] Thomas Bäck, Günter Rudolph, and Hans-Paul Schwefel. 1997. Evolutionary Programming and Evolution Strategies: Similarities and Differences. *Proceedings of the Second Annual Conference on Evolutionary Programming* (04 1997), 11–22.

[10] Philip E Cryer. 2012. Severe hypoglycemia predicts mortality in diabetes. *Diabetes Care* 35, 9 (Sept. 2012), 1814–1816.

[11] Federica Ferrari, Antonio Moretti, and Roberto Federico Villa. 2022. Hyperglycemia in acute ischemic stroke: physiopathological and therapeutic complexity. *Neural Regen. Res.* 17, 2 (Feb. 2022), 292–299.

[12] International Diabetes Foundation. 2024. IDF diabetes atlas 10th edition. https://diabetesatlas.org/data/

[13] Unai Galicia-Garcia, Asier Benito-Vicente, Shifa Jebari, Asier Larrea-Sebal, Haziq Siddiqi, Kepa B Uribe, Helena Ostolaza, and César Martín. 2020. Pathophysiology of type 2 Diabetes Mellitus. *International Journal of Molecular Sciences* 21, 17 (Aug. 2020), 6275.

[14] Erik Hemberg, Jonathan Kelly, and Una-May O'Reilly. 2019. On Domain Knowledge and Novelty to Improve Program Synthesis Performance with Grammatical Evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. ACM, New York, NY, USA, 1039–1046. doi:10.1145/3321707.3321865

[15] J. Ignacio Hidalgo, J. Manuel Colmenar, Gabriel Kronberger, Stephan M. Winkler, Oscar Garnica, and Juan Lanchares. 2017. Data Based Prediction of Blood Glucose Concentrations Using Evolutionary Methods. *Journal of Medical Systems* 41, 9 (2017), 142. doi:10.1007/s10916-017-0788-2

[16] Geoffrey E Hinton and Sam Roweis. 2002. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer (Eds.), Vol. 15. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf

[17] Leon Ingelse, José-Ignacio Hidalgo, José Manuel Colmenar, Nuno Lourenço, and Alcides Fonseca. 2023. Comparing Individual Representations in Grammar-Guided Genetic Programming for Glucose Prediction in People with Diabetes. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 2013–2021. doi:10.1145/3583133.3596315

[18] Leon Ingelse, J. Ignacio Hidalgo, J. Manuel Colmenar, Nuno Lourenço, and Alcides Fonseca. 2024. A comparison of representations in grammar-guided genetic programming in the context of glucose prediction in people with diabetes. *Genetic Programming and Evolvable Machines* 26, 1 (2024), 5. doi:10.1007/s10710-024-09502-5

[19] Hyun-Tae Kim and Chang Wook Ahn. 2015. A New Grammatical Evolution Based on Probabilistic Context-free Grammar. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2*. Springer International Publishing, 1–12. doi:10.1007/978-3-319-13356-0_1

[20] Hyun-Tae Kim, Hyun-Kyu Kang, and Chang Wook Ahn. 2016. A Conditional Dependency Based Probabilistic Model Building Grammatical Evolution. *IEICE Transactions on Information and Systems* E99.D, 7 (2016), 1937–1940. doi:10.1587/transinf.2016edl8004

[21] Kangil Kim, Yin Shan, Xuan Hoai Nguyen, and Robert I McKay. 2013. Probabilistic model building in genetic programming: a critical review. *Genetic Programming and Evolvable Machines* 15, 2 (Sept. 2013), 115–167. doi:10.1007/s10710-013-9205-x

[22] John R Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing* 4, 2 (June 1994), 87–112. doi:10.1007/bf00175355

[23] Roman M. Kozinets, Julia F. Semenova, Vladimir B. Berikov, and Vadim V. Klimontov. 2024. Deep Learning and Machine Learning Models for Predicting Day-Time Glucose in Different Ranges in Patients with Type 1 Diabetes. In *2024 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*. 199–202. doi:10.1109/SIBIRCON63777.2024.10758455

[24] Kui Liu, Linyi Li, Yifei Ma, Jun Jiang, Zhenhua Liu, Zichen Ye, Shuang Liu, Chen Pu, Changsheng Chen, and Yi Wan. 2023. Machine Learning Models for Blood Glucose Level Prediction in Patients With Diabetes Mellitus: Systematic Review and Network Meta-Analysis. *JMIR Medical Informatics* 11 (Nov. 2023), e47833. doi:10.2196/47833

[25] Nuno Lourenço, J. Manuel Colmenar, J. Ignacio Hidalgo, and Óscar Garnica. 2019. Structured Grammatical Evolution for Glucose Prediction in Diabetic Patients. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. ACM, 1250–1257. doi:10.1145/3321707.3321782

[26] Nuno Lourenço, Filipe Assunção, Francisco B Pereira, Ernesto Costa, and Penousal Machado. 2018. Structured Grammatical Evolution: A Dynamic Approach. In *Handbook of Grammatical Evolution*. Springer International Publishing, 137–161. doi:10.1007/978-3-319-78717-6_6

[27] Nuno Lourenço, Francisco B Pereira, and Ernesto Costa. 2016. Unveiling the properties of structured grammatical evolution. *Genetic Programming and Evolvable Machines* 17, 3 (Feb. 2016), 251–289. doi:10.1007/s10710-015-9262-4

[28] Robert I McKay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'Neill. 2010. Grammar-based Genetic Programming: a survey. *Genetic Programming and Evolvable Machines* 11, 3-4 (May 2010), 365–396. doi:10.1007/s10710-010-9109-y

[29] Jessica Mégane, Nuno Lourenço, Penousal Machado, and Dirk Schweim. 2023. The Influence of Probabilistic Grammars on Evolution. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. ACM, 611–614. doi:10.1145/3583133.3590706

[30] Jessica Mégane, Nuno Lourenço, and Penousal Machado. 2021. Probabilistic Grammatical Evolution. In *Genetic Programming: 24th European Conference, EuroGP 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*. Springer International Publishing, 198–213. doi:10.1007/978-3-030-72812-0_13

[31] Jessica Mégane, Nuno Lourenço, and Penousal Machado. 2022. Co-evolutionary probabilistic structured grammatical evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. ACM, 991–999. doi:10.1145/3512290.3528833

[32] Jessica Mégane, Nuno Lourenço, and Penousal Machado. 2022. Probabilistic Structured Grammatical Evolution. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–9. doi:10.1109/cec55065.2022.9870397

[33] Giorgia Nadizar, Luigi Rovito, Andrea De Lorenzo, Eric Medvet, and Marco Virgolin. 2024. An Analysis of the Ingredients for Learning Interpretable Symbolic Regression Models with Human-in-the-loop and Genetic Programming. *ACM Trans. Evol. Learn. Optim.* 4, 1, Article 5 (Feb. 2024), 30 pages. doi:10.1145/3643688

[34] Michael O'Neill, Miguel Nicolau, and Alexandros Agapitos. 2014. Experiments in program synthesis with grammatical evolution: A focus on Integer Sorting. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. 1504–1511. doi:10.1109/CEC.2014.6900578

[35] Alain Ratle and Michèle Sebag. 2002. Avoiding the Bloat with Stochastic Grammar-Based Genetic Programming. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 255–266. doi:10.1007/3-540-46033-0_21

[36] Conor Ryan, John James Collins, and Michael O'Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming: First European Workshop, EuroGP'98 Paris, France, April 14–15, 1998 Proceedings 1*. Springer Berlin Heidelberg, 83–96. doi:10.1007/bfb0055930

[37] Conor Ryan, Michael O'Neill, and JJ Collins. 2018. *Handbook of Grammatical Evolution*. Vol. 1. Springer International Publishing. doi:10.1007/978-3-319-78717-6

[38] R Shan, Robert I McKay, Hussein A Abbass, and Daryl Essam. 2003. Program evolution with explicit learning. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, Vol. 3. IEEE, 1639–1646. doi:10.1109/CEC.2003.1299869

[39] Yin Shan, Robert I McKay, Rohan Baxter, Hussein A Abbass, Daryl Essam, and HX Nguyen. 2004. Grammar model-based program evolution. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, Vol. 1. 478–485. doi:10.1109/CEC.2004.1330895

[40] Yin Shan, Robert I McKay, Daryl Essam, and Hussein A Abbass. 2006. A survey of probabilistic model building genetic programming. In *Scalable optimization via probabilistic modeling*. Springer Berlin Heidelberg, 121–160. doi:10.1007/978-3-540-34954-9_6

[41] Dominik Sobania, Dirk Schweim, and Franz Rothlauf. 2023. A Comprehensive Survey on Program Synthesis With Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 27, 1 (2023), 82–97. doi:10.1109/TEVC.2022.3162324

[42] Léo Françoso Dal Piccol Sotto and Vinícius Veloso de Melo. 2017. A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 1017–1024. doi:10.1145/3071178.3071325

[43] Ivan Tanev. 2005. Incorporating Learning Probabilistic Context-Sensitive Grammar in Genetic Programming for Efficient Evolution and Adaptation of Snakebot. In *Genetic Programming: 8th European Conference, EuroGP 2005. Proceedings 8 (EuroGP'05)*, Maarten Keijzer, Andrea Tettamanzi, Pierre Collet, Jano van Hemert, and Marco Tomassini (Eds.). Springer-Verlag, Berlin, Heidelberg, 155–166. doi:10.1007/978-3-540-31989-4_14

[44] P.A. Whigham. 1995. Inductive bias and genetic programming. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. 461–466. doi:10.1049/cp:19951092

[45] Peter A Whigham. 1995. Grammatically-based Genetic Programming. In *Proceedings of the workshop on genetic programming: from theory to real-world applications*, Vol. 16. 33–41.

[46] Pak-Kan Wong, Leung-Yau Lo, Man-Leung Wong, and Kwong-Sak Leung. 2014. Grammar-Based Genetic Programming with Bayesian network. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 739–746. doi:10.1109/cec.2014.6900423

[47] Pak-Kan Wong, Man-Leung Wong, and Kwong-Sak Leung. 2019. Probabilistic grammar-based deep neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 87–88. doi:10.1145/3319619.3326778