Genetic Programming and Evolvable Machines (2025) 26:27
https://doi.org/10.1007/s10710-025-09524-7

Semantic-based recombination and mutation in cellular-inspired genetic programming

Luigi Rovito[1] · Lorenzo Bonin[1] · Davide Farinati[3] · Leonardo Vanneschi[3] · Luca Manzoni[1] · Andrea De Lorenzo[2] · Gloria Pietropolli[1]

Received: 18 April 2025 / Revised: 29 July 2025 / Accepted: 28 September 2025

Abstract
The incorporation of a Cellular Automata (CA)-like structure into the population of Evolutionary Algorithms (EAs) has been shown to enhance solution quality. However, research on CA-like structures in the context of Genetic Programming (GP) remains limited. This work examines the impact of introducing such structures in Geometric Semantic variants of GP, specifically focusing on the well-established Geometric Semantic GP (GSGP) and the recently proposed SLIM-GSGP, which prioritizes generating smaller and more interpretable individuals. Furthermore, we analyze how cellular structures influence the effectiveness of semantic-based recombination and mutation in both GSGP and SLIM-GSGP. To this end, we conduct a comprehensive evaluation of these genetic operators, examining their effects both individually and in combination. We provide insights into how CA-like structures and semantic genetic operators influence both the quality and size of solutions in GSGP and SLIM-GSGP, offering a clear understanding of the trade-offs associated with these approaches.

Keywords Evolutionary algorithms · Genetic programming · Geometric semantic genetic programming · Cellular automata · Symbolic regression · SLIM geometric semantic genetic programming

1 Introduction

Cellular Automata (CA) [1–3] are classical nature-inspired computing models consisting of n-dimensional grids of cells, each capable of occupying one of a finite set of states. Each cell interacts within a defined neighborhood, and at each discrete time step, all cell states are synchronously updated according to a uniform local rule.

Extended author information available on the last page of the article

Published online: 19 November 2025

The inherent properties of CAs make them a promising model for evolution in Evolutionary Algorithms (EAs). Representing the population as a toroidal structure restricts interactions to individuals within the same neighborhoods [4–6], which can provide several algorithm-dependent benefits. By limiting interactions, this structure mitigates premature convergence to local optima and promotes diversity in the search process, often leading to the discovery of higher-quality solutions [4, 7–12].

In this context, the problem of Symbolic Regression (SR) has attracted significant attention, with numerous techniques proposed to discover accurate mathematical expressions that interpolate available data [13, 14], despite its inherent difficulty [15].

The combination of CAs and EAs has shown promise [13, 14]; however, integrating Genetic Programming (GP) with spatial structures such as CA remains largely unexplored, particularly for SR. Most existing research on CA-based GP has either focused on domains other than SR [16–18] or pursued parallel implementations aimed at enhancing computational efficiency [19]. Furthermore, these studies have not been assessed using modern, widely recognized SR benchmarks [14, 20].

Additionally, semantic-based variations of GP, such as Geometric Semantic Genetic Programming (GSGP) [21, 22], have received little attention in CA-inspired frameworks. GSGP is a GP variant that explores the space of programs by using their semantic representations and by performing modifications that have direct effect on the semantic of the evolved programs.

Only recently a cellular structure has been introduced to regulate the spread of dominant individuals and mitigate premature convergence in GSGP [7]. However, this approach does not address the inherent exponential growth of trees in GSGP. A promising solution to this issue is Semantic Learning algorithm based on Inflate and deflate Mutation (SLIM) [23, 24], a recent GSGP variant that employs a modified Geometric Semantic Mutation (GSM) operator to generate more compact offspring, thus reducing model size. This method was firstly developed to only use mutation without any type of recombination, thus limiting the exploration capabilities of the algorithm and focusing on exploitation only. However, in [25], Pietropolli et al. integrated several types of crossover operators within SLIM, showing that this technique can be also employed with both recombination and mutation together to provide a higher-quality search space exploration.

Building on the insights from [7], Rovito et al. [26] explored the integration of cellular structures into various GP approaches by evaluating two primary aspects: model performance, to measure solution quality, and model size. As demonstrated in [23, 24], smaller models enhance interpretability by yielding more comprehensible mathematical expressions, making them more suitable for analysis with explainability techniques. Compact models also offer practical benefits, including lower computational and memory demands, and greater ease of deployment and monitoring in real-world applications.

The combination of GP and GSGP variants with spatial structures introduces non-trivial interactions that affect selection pressure, diversity maintenance, and solution complexity. Theoretical modeling of such hybrid systems is limited, as existing analysis often assume populations where individuals can interact freely. Moreover, the overlapping effects of each technique on performance metrics complicate interpretation.

This manuscript extends our previous work [26], where we introduced cellular structures into three variants of Genetic Programming: standard GP, GSGP, and SLIM. We focus on these three GP algorithms in their standard versions to ensure a clear comparison, as they each possess distinct strengths and limitations. We performed a systematic comparison of their performance under mutation-only settings. In this extended version, we investigate the role of semantic crossover in GSGP and SLIM within a cellular framework. Specifically, we assess whether the introduction of crossover alters the trade-offs between accuracy and model size observed in the mutation-only case, and how the interaction between crossover and cellular constraints impact evolutionary dynamics.

Our study begins with a thorough literature review that consolidates previous research on GP techniques and cellular frameworks (Sect. 2). We then conduct an extensive experimental analysis, evaluating model performance, model size, and diversity according to different cellular structures and genetic operators combinations.

Our goal is to comprehensively understand how toroidal cellular grids and genetic operators affect the population dynamics and the performance of core GP algorithms. Additionally, we aim to clarify the balance between predictive accuracy and model size that can be achieved by integrating GP techniques with CA-inspired structures, by also analyzing the impact of the individual semantic-based genetic operators.

Finally, although the topic may seem specific, our ablation study includes various configurations with and without cellular automata and semantic operators, enabling broader comparisons. While we cannot say with certainty that our conclusions extend to all GP methods, the set of algorithms and experiments we include covers a wide range of potential GP variants. We therefore believe our findings generalize to a broad set of use cases.

In Sect. 2 we discuss the literature review concerning cellular structures applied to EAs, in Sect. 3 we outline the main GP variants analyzed, the genetic operators under examination, and the cellular-based selection procedure we employ, in Sect. 4 we describe our experimental methodology and setting, in Sect. 5 we show the outcomes of our analysis and we discuss the main findings, and in Sect. 6 we summarize our conclusions and outline possible future research directions.

2 Related works

Prior studies have investigated the integration of spatial structures in EAs [4–6]. This approach restricts interactions to localized subsets of the population, known as neighborhoods, which are typically arranged in a grid. In Genetic Algorithms (GAs) [27], a Cellular Genetic Algorithm (cGA) [4–6] is used to introduce spatial structuring, and research [4, 6] has examined the effect of different neighborhood configurations on performance. Deng et al. [28] leveraged cGA to improve optimization in problems like the traveling salesman problem, where cGA was combined with simulated annealing [29]. Alba et al. [4] introduced a dynamic version of cGA, where exploration-exploitation is adjusted during evolution. In another work, Murata et al. [8] implemented C-MOGA, a GA incorporating cellular structuring for local selection in multi-objective optimization. Building on this idea, Nebro et al. [9] later proposed

MOCell, drawing inspiration from traditional cGA. Additionally, Mariot et al. [30] utilized GA and GP with CAs to design orthogonal Latin squares. Beyond GA, other EAs have also leveraged cellular structures to regulate the spread of solutions within a population [10, 31, 32]. This demonstrates that applying a cellular structure to the population is generally feasible regardless of the specific algorithm used.

GP [33] has demonstrated its versatility and effectiveness in solving a wide range of problems by evolving computer programs [34–39]. A key advantage of its learning process and solution representation is the potential to uncover interpretable models [40–49]. Folino et al. [19] proposed a scalable parallel implementation of GP based on a cellular structure, incorporating load balancing to evenly distribute tasks across processors. Their results showed that this approach can outperform both traditional GP and the island model [50], where evolutionary runs occur independently on separate population subsets. Moreover, they proposed a parallel cellular-based implementation of GP to tackle classification problems [16]. In later studies, Takac proposed a cellular-based GP method for both classification [17] and data mining [18]. Furthermore, research by [51] and [52] showed that integrating spatial population structure with local elitist replacement effectively mitigates bloat (unnecessary growth in tree size) in GP, while preserving performance.

Biodiversity in a population can be maintained by applying speciation, in which individuals are grouped into distinct niches based on structural similarities, restricting crossover to individuals within the same niche. Della Cioppa et al. [53] proposed an adaptive species discovery strategy to overcome the limitations of traditional niching methods, which often depend on prior knowledge of the fitness landscape. Building on the NEAT algorithm [54], Trujillo et al. [55] applied speciation to control program growth in GP through neat-GP, which promotes complexity only when necessary by dividing the population into species based on size and structure. Juarez et al. [56] improved neat-GP by incorporating a local search operator to enhance solution quality. Cussat et al. [57] proposed a network distance metric to speciate populations of artificial gene regulatory networks, showing that speciation fosters diversity and maintains smaller individuals. Martins et al. [58] combined GAs with speciation and grid pattern recognition to reduce investment risks and increase profits. Wickman et al. [59] applied speciation in Reinforcement Learning (RL) to evolve diverse policies, while Pietropolli et al. [12] proposed using substrates with empty cells (barriers) instead of a flat toroidal grid to slow genetic propagation and enhance diversity.

Various studies have examined the impact of selection pressure and sampling strategies in both standard EAs [60, 61] and cellular-based EAs [11, 62]. In cellular-based EAs, spatial structure plays a crucial role in regulating takeover time, i.e., the number of generations required for a dominant individual to spread throughout the population. A longer takeover time is associated with greater diversity [11].

GSGP [21, 22], which has beeen shown to outperform GP in various tasks [13, 14, 21, 63–67], also suffers from notable limitations. These are mainly due to the susceptibility of GSGP to local optima, where evolution may stagnate, and its tendency toward premature convergence, leading to low takeover time [68]. It is common to employ only a mutation based GSGP, since it was proven that the Geometric Semantic Crossover (GSC) is not an effective operator, due to the fact that it constrains the evolution inside the convex hull of the population, limiting exploration [69]. To

address these issues, Bonin et al. [7] introduced Cellular Geometric Semantic Genetic Programming (cGSGP), a variant of GSGP that imposes a cellular toroidal structure to the population. Their findings indicate that this approach enhances diversity in the early stages of evolution by increasing takeover time, ultimately leading to improved performance over standard GSGP.

While these efforts help limit premature convergence, GSGP still suffers from exponential growth of the trees across the generations. Given the extensive research on improving this technique [70–75] and the availability of efficient implementations [76, 77], numerous studies have sought to address bloating and mitigate the uncontrolled growth of trees size [67, 72, 75, 77–80]. However, these methods still rely on Geometric Semantic Operators (GSOs), which inherently generate offspring larger than their parents [81].

Recently, Vanneschi et al. [23] proposed SLIM, a novel variant of GSGP that employs a specialized GSM, capable of producing smaller offspring without compromising performance. This advancement marks a significant shift in how GSGP solutions are generated and utilized. Building on this foundation, Pietropolli et al. [25] expanded SLIM by incorporating various crossover operators, demonstrating that this technique, originally designed to rely solely on mutation, can effectively integrate both recombination and mutation. This enhancement enables a more diverse and higher-quality exploration of the search space. Despite advancements in controlling bloat, the interaction between SLIM and spatial structuring remains underexplored. In particular, combining semantic-driven bloat control with spatial structuring could address complementary challenges in GSGP-based methods.

3 Methods

In this section, we briefly describe GP [33, 82] and its two variants [21, 24] that we will investigate alongside vanilla GP. We also describe the cellular-based selection strategy adopted in [7].

3.1 Genetic programming

GP is an evolutionary algorithm that evolves computer programs to solve problems without requiring explicit programming. It is derived from the broader field of GA and operates by mimicking natural selection, iteratively refining a population of candidate solutions. This approach is particularly useful for addressing complex, nonlinear problems where the structure of the solution is unknown or difficult to define beforehand. The fundamental mechanisms of GP include selection, which prioritizes the best individuals for reproduction, crossover (combining solutions), and mutation, which introduces variation to maintain diversity in the population.

A notable application of genetic programming is symbolic regression, a technique for discovering the mathematical expression that best models the relationship between input variables and outputs. Unlike conventional regression methods that require predefined model structures (e.g., linear or polynomial models), symbolic regression autonomously determines both the equation form and its parameters. This

makes it highly adaptable and capable of capturing intricate, non-linear relationships within data. The resulting models are often interpretable, providing insights into underlying systems in various fields such as physics, biology, or economics.

Symbolic regression via GP has been successfully applied to system identification, financial modeling, bioinformatics, and engineering optimization. However, its effectiveness relies on careful configuration, including selecting the right function set, defining fitness evaluation, and managing program size growth (bloat). Recent advancements, such as hybrid methods and machine learning integration, continue to improve the efficiency and effectiveness of genetic programming, solidifying its role as a tool for predictive modeling and knowledge discovery [83].

Genetic programming typically represents individuals as tree structures, as illustrated in Fig. 1.

For an input vector $X = [x_1, x_2, \ldots, x_n]$, an individual's predictions $S = [T(x_1), T(x_2), \ldots, T(x_n)]$ can be referred to as its semantics [21]. While standard GP operations like one-point crossover [83] and one-point mutation [83] visibly alter an individual's structure, their semantic impact remains uncertain until evaluation.

3.2 Geometric semantic genetic programming

GSGP [21] is a variant of GP that replaces traditional genetic operators with GSOs. GSOs uniquely map genetic changes to predictable shifts in semantic space. These operators generate unimodal error surfaces for supervised learning tasks, as discussed in [22, 84].

Recent research [67, 69] suggests that mutation-based GSGP can match or exceed crossover-based performance. The GSM operator for symbolic regression, introduced in [21], is defined as follows:

GSM. Given a function $T : \mathbb{R}^n \to \mathbb{R}$, GSM with mutation step ms creates a new function: $GSM(T) = T + \mathrm{ms} \cdot (T_{R1} - T_{R2})$, where T_{R1} and T_{R2} are random functions, and ms controls mutation magnitude.

This approach perturbs each semantic component of T uniquely. The difference between two random functions, T_{R1} and T_{R2}, ensures a zero-centered distribution, balancing positive and negative perturbations.

Research [65, 67, 85] demonstrates that constraining T_{R1} and T_{R2}'s output range (e.g., [0, 1] via sigmoid functions [85]) enhances GSGP's generalization. This ensures perturbations remain within $[-\mathrm{ms}, \mathrm{ms}]$, effectively performing ball mutations in semantic space [21, 67, 84]. However, GSM increases individual size, as the resulting function embeds the parent alongside additional random expressions.

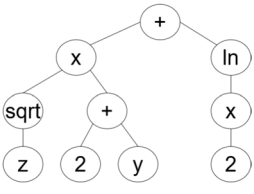

Fig. 1 Example of an individual in GP

3.3 Alternative ball mutation definitions

The GSM operation can be rewritten as: $GSM(T) = T + \Delta(T)$, where $\Delta(T)$ represents the mutation-induced perturbation.

If T_{R1} and T_{R2} are confined to [0, 1] via sigmoid transformation, then: $\Delta(T) = 2\text{SIG} = \text{ms} \cdot (S(T_{R1}) - S(T_{R2}))$, where S is the sigmoid function.

Other perturbation methods exist [23, 24, 86]:

- $\Delta(T) = 1\text{SIG} = \text{ms} \cdot (2S(T_R) - 1)$, using a single random function T_R.
- $\Delta(T) = \text{ABS} = \text{ms} \cdot (1 - 2/(1 + |T_R|))$.

1SIG and ABS require fewer computations than 2SIG [24].

Another approach scales the parent instead of adding perturbation: $GSM(T) = T \cdot (1 + \Delta(T))$.

Combining three perturbation functions (2SIG, 1SIG, ABS) with two mutation methods (addition, scaling) produces six variants. For instance, three approaches can be implemented by using scaling as mutation method [23, 24]:

- SLIM*2SIG, defined as: $GSM(T) = T \cdot (1 + 2SIG)$.
- SLIM*1SIG, defined as: $GSM(T) = T \cdot (1 + 1SIG)$.
- SLIM*ABS, defined as: $GSM(T) = T \cdot (1 + ABS)$.

3.4 Semantic learning algorithm based on inflate and deflate mutation

SLIM [23, 24] is an extension of GSGP that, alongside the geometric semantic mutation described earlier, introduces a novel mutation operator. Similarly to the previously described mutation, this new operator induces a ball mutation of radius ms on the semantic space. However, unlike the former operator, this new mutation produces offspring that are smaller in size than their parents. Consequently, this new operator is named Deflate Geometric Semantic Mutation (Deflate Geometric Semantic Mutation (DGSM)), while the prior operator will be referred to as Inflate Geometric Semantic Mutation (Inflate Geometric Semantic Mutation (IGSM)).

The inspiration behind DGSM stems from two key observations. First, the definition of GSM can be rewritten as $\text{GSM}(T) = T + ms \cdot (T_{R1} - T_{R2}) = T - ms \cdot (T_{R2} - T_{R1})$. Second, since the random expressions T_{R1} and T_{R2} are sampled independently from the same distribution, their roles are interchangeable. Consequently, GSM can be equivalently defined as: $\text{GSM}\,(T) = T - \text{ms} \cdot (T_{R1} - T_{R2})$.

This observation is key to developing a mutation operator capable of reducing the size of individuals. For example, applying GSM three times on an individual T yields: $T_M = GSM^3(T) = T + \text{ms} \cdot (T_{R1} - T_{R2}) + \text{ms} \cdot (T_{R3} - T_{R4}) + \text{ms} \cdot (T_{R5} - T_{R6})$

Applying GSM one more time with subtraction instead of addition results in: $T_{M'} = T + \text{ms} \cdot (T_{R1} - T_{R2}) + \text{ms} \cdot (T_{R3} - T_{R4}) + \text{ms} \cdot (T_{R5} - T_{R6}) - \text{ms} \cdot (T_{R7} - T_{R8})$

A common practice in literature is to reuse random expressions [67]. By reusing T_{R3} and T_{R4} in place of T_{R7} and T_{R8} when a new mutation is applied, the result becomes:

$$\begin{aligned} T_{M'} =& T + \text{ms} \cdot (T_{R1} - T_{R2}) + \cancel{\text{ms} \cdot (T_{R3} - T_{R4})} \\ &+ \text{ms} \cdot (T_{R5} - T_{R6}) - \cancel{\text{ms} \cdot (T_{R3} - T_{R4})} \end{aligned} \tag{1}$$

$$= T + \text{ms} \cdot (T_{R1} - T_{R2}) + \text{ms} \cdot (T_{R5} - T_{R6}) \tag{2}$$

This simplification results in an individual with a smaller genotype than T_M. The DGSM operator utilizes this principle by eliminating one of the elements previously added by IGSM. Importantly, this simplification does not constitute a backtracking step, as the resulting individual was never present in the previous population.

In the implementation of SLIM, the genotype of each individual can be visualized as a *linked list*, where the first node represents the initial random expression, and each subsequent node corresponds to an expression added by IGSM. This structure is illustrated in Fig. 2.

DGSM removes one non-initial element from the linked list, simplifying the genotype.

While this example employs the SLIM+SIG2 variant, the same rationale applies to the other variants. For SLIM*ABS, the deflate mutation is implemented by dividing instead of subtracting, leading to further simplification.

3.4.1 Crossover in SLIM

In SLIM, the linked-list representation not only facilitates the implementation of the deflate operator but also enables novel approaches for recombining genetic material within the population. Standard GSGP faces well-documented challenges: (1) Even more than GSM, GSC leads to exponential growth in population size due to its tendency to produce offspring larger than their parents [21], and (2) GSC has limited search capability, as it cannot generate individuals outside the convex hull of the existing population [87, 88].

To mitigate these limitations, [25] introduced new crossover methods that address both constraints. For (1), these methods were designed to control individual size, preventing the rapid growth typical of GSC. For (2), they extended beyond standard geometricity–unlike GSC, which generates offspring along the segment between parents, the new crossovers allow for a broader search space while maintaining a clear geometric interpretation.

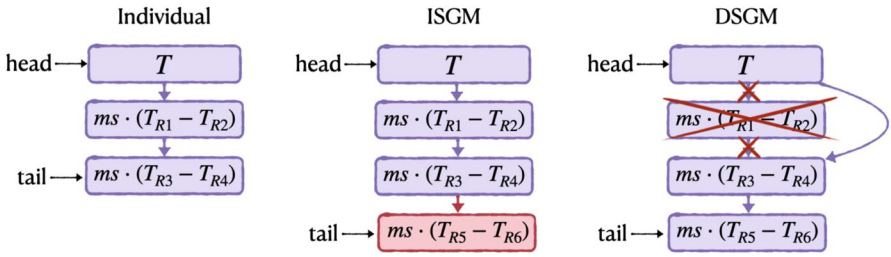

Fig. 2 An example of a SLIM+2SIG individual, visually depicting the effects of IGSM and DGSM on its genotype. The IGSM appends new blocks, while DGSM removes them

Among the different operators proposed in [25], the authors introduced Swap Crossover (XOSw), a crossover that swaps elements within the linked list between parents, enabling genetic recombination while preserving offspring size. XOSw is defined as:

Swap Crossover (XOSw) Given two parent functions $\mathcal{T}_1 = (T_1, \Delta T_1^{(1)}, \ldots, \Delta T_{n_1}^{(1)})$ and $\mathcal{T}_2 = (T_2, \Delta T_1^{(2)}, \ldots, \Delta T_{n_2}^{(2)})$, where $n_1 < n_2$ without loss of generality, XOSw produces two offspring functions:

$$\begin{aligned} \mathcal{T}_{\text{XOSw}_1} &= \text{sel}(T_1, T_2) \cdot \textstyle\prod_{i=1}^{n_1} \text{sel}(\Delta T_i^{(1)}, \Delta T_i^{(2)}) \cdot \prod_{i=n_1+1}^{n_2} \text{sel}(1, \Delta T_i^{(2)}), \\ \mathcal{T}_{\text{XOSw}_2} &= \overline{\text{sel}}(T_1, T_2) \cdot \textstyle\prod_{i=1}^{n_1} \overline{\text{sel}}(\Delta T_i^{(1)}, \Delta T_i^{(2)}) \cdot \prod_{i=n_1+1}^{n_2} \overline{\text{sel}}(1, \Delta T_i^{(2)}) \end{aligned} \tag{3}$$

where Π denotes multiplication over the terms, $\text{sel}(a, b)$ selects a or b with equal probability, with 1 serving as the neutral multiplicative element. The operator $\overline{\text{sel}}(a, b)$ returns the complement of $\text{sel}(a, b)$, such that if $\text{sel}(a, b) = a$, then $\overline{\text{sel}}(a, b) = b$, and vice versa.

This crossover generates two distinct individuals. For the first offspring, each block is randomly selected from the corresponding position in either parent with equal probability. The second offspring is then constructed by taking all blocks not selected by the first; for each position, if the first offspring inherits a block from one parent, the second offspring inherits the corresponding block from the other parent. When parents differ in length, any extra blocks from the longer parent (for $i = n_1 + 1, \ldots, n_2$) may or not be assigned to the first offspring at random. The second offspring then inherits any remaining unassigned blocks, ensuring distinct contributions without overlap.

3.5 Cellular methods

Standard GP and GSGP algorithms lack spatial structure within the population. Selection is performed across the entire population and crossover is potentially allowed between any pair of individuals.

In algorithms inspired by CA [1–3], a spatial structure is imposed, leading to a notion of neighborhood within the population. CA is a formal model widely studied in computability theory, consisting of finite automata arranged based on a specific topology, with each cell updating its internal state according to a local rule that depends exclusively upon a given number of neighboring automata. Just like in [7], individuals in the population are arranged according to a toroidal grid, where each individual belongs to a specific cell (i.e., a spatial location), randomly chosen at initialization, which, in turn, belongs to a specific neighborhood. With this configuration, we impose a spatial structure on the population, and, consequently, a notion of neighborhood that can be leveraged in the selection phase.

The neighborhood of radius r for a given individual is defined by a hypercube with side $2r + 1$, corresponding to the Moore neighborhood centered on the cell in the grid containing the individual at hand. Note that in this configuration each individual belongs to its own neighborhood. The toroidal configuration of the grid ensures all cells have complete neighborhoods, otherwise, cells near the hypercube faces would

be incomplete. We denote the n-dimensional toroidal grid with radius $r \in \mathbb{N}$ as $\mathcal{T}_r^n$, while we denote the absence of toroidal configuration over the population as $\mathcal{T}^0$. Note that there are no disconnected neighborhoods in $\mathcal{T}_r^n$ when $r > 0$. This means that information can be exchanged between any two cells within the grid. In other words, there are no isolated or independent sub-populations.

After defining a neighborhood-based toroidal structure, we detail how this configuration is influencing the selection process and the choice of the parents for the crossover operator. Mutation, being inherently local, does not need any modification. To this end, we adopt the aforementioned described cellular-inspired structure along with the selection strategy in [7], which allows only local interactions between individuals.

Given an *n*-dimensional toroidal grid and an individual T_i in a cell *i* of the grid, we denote $\mathcal{N}_i$ as the neighborhood centered in the cell *i*. For instance, with two dimensions, we have that $i = (i_1, i_2)$ and a neighborhood of radius *r* is defined as $\mathcal{N}_i = \{j = (j_1, j_2) \text{ such that } ||i - j||_\infty \leq r\}$, where $||x||_\infty$ is the maximum norm $||x||_\infty = \max_{1 \leq i \leq n} |x_i|$. An example of toroidal grid with neighborhoods of radius one is depicted in Fig. 3.

T_i is replaced by a new tree derived from others in its neighborhood $\mathcal{N}_i$ and selection is performed for each cell in the grid. Specifically, for each cell *i*, the two individuals chosen for crossover are selected among the individuals within the same neighborhood $\mathcal{N}_i$. If no crossover is performed, T_i is replaced with another (mutated) individual in $\mathcal{N}_i$.

This approach (defined as TRS_p) is a tournament-based selection within local neighborhoods, where the selection pressure is defined by a value $p \in (0, 1]$ rep-

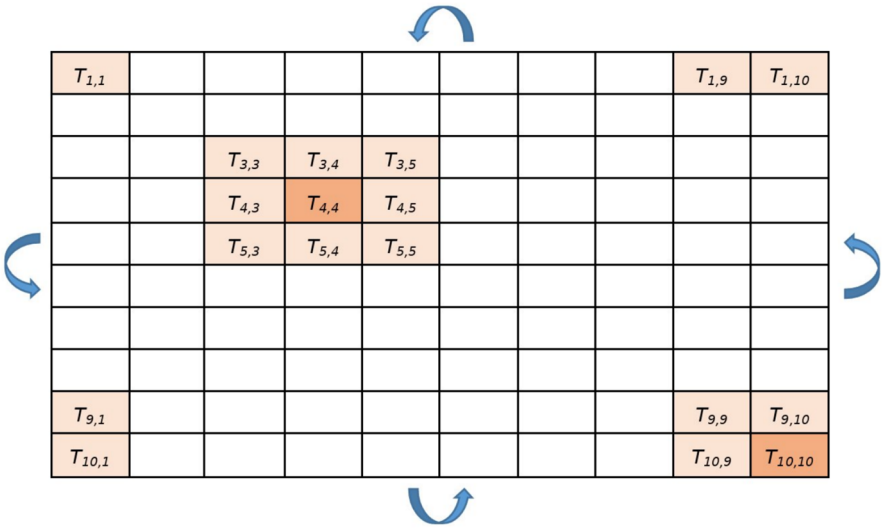

Fig. 3 Example of a population with 100 individuals distributed according to $\mathcal{T}_1^2$. In the figure, two positions with the corresponding neighborhoods are highlighted, following the toroidal structure when the position is close to the borders. Each cell contains the individual in that position, whose row index and column index are indicated as a subscript

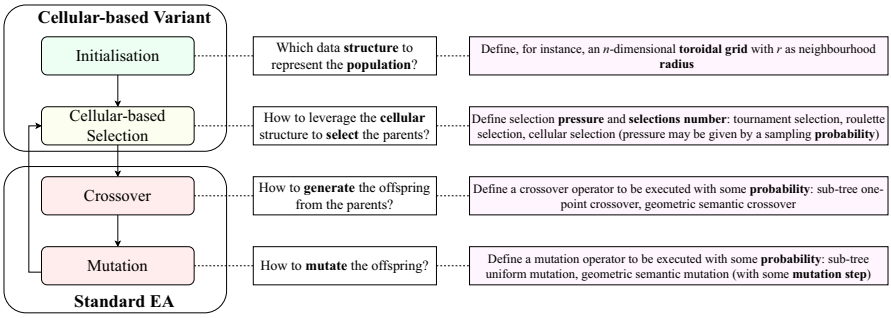

Fig. 4 Diagram describing a baseline cellular-based evolutionary algorithm. For each activity, the main design choice is highlighted, and an overview of the main hyper-parameters and variants is presented. Especially, we highlight in bold the hyper-parameters and the main building blocks that compose the design choices

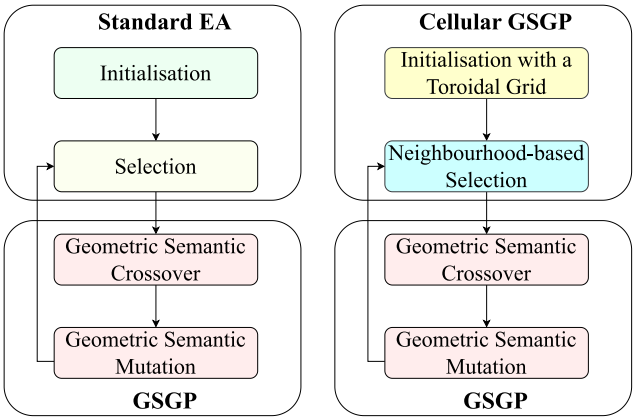

Fig. 5 Diagram comparing a vanilla GSGP and cGSGP. The figure highlights which are the components that differ between the two techniques

resenting a proportion of the given neighborhood. A subset $\mathcal{S}$ of $\mathcal{N}_i$ is sampled by iterating across the elements of $\mathcal{N}_i$. For each element, it is inserted into $\mathcal{S}$ with probability p ($\mathcal{S}$ is initialized as an empty set prior to the beginning of the iterations). Then, a rank-based selection is applied on $\mathcal{S}$: if no crossover is performed, then the new individual in cell i will be a mutated version of the best individual in $\mathcal{S}$. If crossover is performed, then the new individual will be a tree generated by performing crossover between the two best individuals in $\mathcal{S}$. Figure 4 shows a diagram describing how cellular structures can be integrated within an evolutionary algorithm. Figure 5 shows a diagram comparing vanilla GSGP and cellular-based GSGP.

4 Experimental methodology

In this section, we describe our experimental methodology and setting.

To guide our experimental design, we formulate the following research questions:

RQ1 *How does the introduction of a cellular (toroidal) structure affect the performance and size of evolved solutions in different GP algorithms?*

RQ2 *How do different exploration pipelines (i.e., combinations of mutation and crossover) influence accuracy and model size, especially in semantic methods such as GSGP and SLIM?*

RQ3 *Which combinations of algorithm, exploration pipeline, and spatial structure achieve the best trade-off between accuracy and interpretability (measured by model size)?*

RQ4 *To what extent do cellular structures promote semantic diversity during evolution, and how does this affect search dynamics?*

We set a population size of 100 that is evolved for 1000 generations with elitism enforced (the best solution according to the training set is preserved to the next generation). In case of standard non-cellular ($\mathcal{T}^0$) algorithms, we employ tournament selection with a tournament size of 4 as selection criterion. In case of cellular algorithms, we use a 10x10 bi-dimensional toroidal grid. We test cellular methods with $p = 1$ and radius between 2 ($\mathcal{T}_2^2$) and 3 ($\mathcal{T}_3^2$) since, according to [7], these are the hyper-parameter values that bring more consistent results.

We use ramped half-and-half tree initialization [33, 82, 89], with an initial maximum depth of 6. The function set consists of the following operators: $+, -, \times, \div^*$, where * indicates protected division to avoid numerical errors (if the denominator is zero, then one is returned). The terminal set consists of the variables of the problem. As in [23, 24], we do not explicitly add ephemeral constants to the terminal set.

Together with different cellular structures, we also test different combinations of genetic operators. Specifically, in case of the semantic variants of GP (GSGP and SLIM), we test each algorithm with only crossover ($\mathcal{E}_{\text{cx}}$), only mutation ($\mathcal{E}_{\text{mut}}$), and both crossover and mutation ($\mathcal{E}_{\text{cx,mut}}$) executed in a mutually exclusive way for each individual in the population and for each generation. From now on, we refer to these three combinations of genetic operators as exploration pipelines. In our experimental campaign, we only run vanilla GP with $\mathcal{E}_{\text{cx,mut}}$ as this is the standard algorithm. In case of SLIM, we omit the results for $\mathcal{E}_{\text{cx}}$ since, considering the crossover operator under employment, without executing mutation the population does not change over time, making this specific case of variation useless.

GP performs sub-tree crossover with probability 0.8 and sub-tree mutation with probability 0.2. When we use $\mathcal{E}_{\text{cx,mut}}$ as exploration pipeline, GSGP performs GSM with probability 0.8 and GSC with probability 0.2, while, analogously, SLIM performs SLIM mutation with probability 0.8 and SLIM crossover with probability 0.2. Following [23], we set SLIM inflate and deflate probabilities to 0.3 and 0.7, respectively. The mutation step *ms* is sampled uniformly from the interval [0, 1] for each mutation event in both GSGP and SLIM.

Regarding the specific SLIM mutations employed, we focus our analysis on $\text{SLIM}^+_{\text{ABS}}$, $\text{SLIM}^+_{\text{SIG1}}$, and $\text{SLIM}^+_{\text{SIG2}}$, which are the SLIM versions on which cellular structures have been shown to be more effective [26]. Regarding the SLIM crossover, we employ the swap crossover since, according to [25], it has been shown to be the most effective one. The choice of crossover and mutation probabilities when using $\mathcal{E}_{\text{cx,mut}}$ is justified by the findings in [25], where Pietropolli et al. tested dif-

ferent crossover/mutation probabilities and found out that no meaningful difference is recorded when slightly varying those probabilities. Therefore, we adopt standard values for these probabilities.

We adopt six datasets that are commonly used in GP tasks [20, 65, 67, 74] (Table 1).

Each dataset is randomly split into 30 train-test partitions using a 7:3 ratio. For every method, dataset, and hyperparameter configuration, we conduct 30 independent runs, corresponding to each partition.

Our experiments focus on three key metrics: (i) the Root Mean Squared Error (RMSE) [94], which quantifies model accuracy, with lower values indicating better regression performance; (ii) Model size ($\log_{10}(\ell)$), defined as the base-10 logarithm of the total number of nodes (ℓ), serving as an indicator of interpretability, memory usage, and computational complexity, as discussed in [23]; (iii) Global Moran's I (I) [95, 96], a spatial autocorrelation metric [97] that captures diversity within neighborhood-based populations.

I assesses the similarity (or dissimilarity) of values located in neighboring locations. Provided that an individual is represented by its semantics, we have:

$$I = \frac{M}{W} \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} \mathrm{w}_{ij} (\mathrm{x}^i - \bar{\mathrm{x}})(\mathrm{x}^j - \bar{\mathrm{x}})}{\sum_{i=1}^{M} (\mathrm{x}^i - \bar{\mathrm{x}})^2}$$

where M is the population size, $\mathrm{w}_{ij} \in \mathbb{R}$ is the value located at the i-th row and the j-th column of the matrix $\mathrm{w} \in \mathbb{R}^{M \times M}$, $\mathrm{w}_{ii} = 0$ for $1 \leq i \leq M$, $W = \sum_{i=1}^{M} \sum_{j=1}^{M} \mathrm{w}_{ij}$, x^i is the i-th individual in the population represented as a semantic vector, and $\bar{\mathrm{x}}$ is the mean of all the semantic vectors of the individuals in the population. In our case, $\mathrm{w}_{ij} = 1$ if $i \neq j$ and the j-th individual belongs to the neighborhood of the i-th individual (in a standard non-cellular method the entire population is a single neighborhood), otherwise $\mathrm{w}_{ij} = 0$.

The I measure ranges from -1 to 1. Values near 1 indicate strong positive spatial autocorrelation, meaning that individuals with similar semantic content are clustered together within the population. Conversely, values approaching -1 suggest negative spatial autocorrelation, where similar individuals are dispersed rather than grouped. A I value close to 0 implies that the spatial distribution of individuals is indistinguishable from a random arrangement.

A strictly positive I value indicates that each neighborhood forms a cluster of similar individuals, while remaining distinct from other neighborhoods. This spatial structuring promotes diversity by ensuring that different regions of the search space are explored independently. This is an effect reinforced by the toroidal grid.

Table 1 Summary of the datasets used in the experiments

Dataset	Records	Variables	References
Airfoil (ARF)	1502	5	[90]
Concrete (CNC)	1029	8	[63]
Slump (SLM)	102	9	[91]
Yacht (YCH)	307	6	[92]
Parkinson (PRK)	5875	18	[64]
QSAR aquatic toxicity (QSR)	546	8	[93]

For clarity, we use the term "*algorithm*" to denote different GP approaches, specifically GSGP and SLIM, while "*method*" refers to the different variants in the same group when examined together with statistical tests, where a variant is a combination of algorithm, cellular structure, and exploration pipeline. Our Python implementation, `cslim`, is openly accessible online.[1]

5 Results and discussion

In this section, we show the results of our experimental phase and discuss the main outcomes.

5.1 Statistical analysis and convergence rates

We start our experimental analysis by investigating the influence of different cellular structures and genetic operators on the performance of a given semantic EA, measured in terms of RMSE and $\log_{10}(\ell)$.

We recall that the term $\mathcal{T}^0$ refers to the non-cellular standard version, while $\mathcal{T}_2^2$ and $\mathcal{T}_3^2$ indicate, respectively, cellular methods with radius 2 and 3. Moreover, we use $\mathcal{E}_{\text{cx}}$ to denote methods that only perform crossover, $\mathcal{E}_{\text{mut}}$ for methods that only perform mutation, and $\mathcal{E}_{\text{cx,mut}}$ for methods performing both crossover and mutation in a mutually exclusive way for each individual.

This first analysis focused on comparing the different variants by inspecting three analysis dimensions: the algorithm, the cellular structure, and the exploration pipeline. Given the *k*-th dimension, we fix the other two dimensions and we compare the different variants obtained by varying the *k*-th dimension by using statistical tests. We perform this step for each of the three analysis dimensions.

Statistical comparisons are executed by using the Wilcoxon-Mann–Whitney test [98] with $\alpha = 0.05$. When multiple variants are compared, *p*-values are adjusted by the Holm-Bonferroni correction [99] (these tests are performed after assessing whether a preliminary multi-group Kruskal-Wallis test [100] is passed). In the following tables (Tables 2, 3 and 4), for each dataset and variants group, the eventual statistically superior variant among the compared ones is marked with an asterisk (*), and the one with lowest median value is indicated in bold. Additionally, variants that outperform at least another variant in the same group according to the pair-wise test are marked with a plus ($^+$).

In the subsequent tables, we present the results related to the most meaningful variants and configurations. The complete versions of these tables including all the possible comparisons are available in the Appendix section (Sect. 7).

We further analyze performance and size by presenting the median test fitness evolution of the best individual over 30 runs in Fig. 6 and the median size evolution of the best individual in Fig. 7. Since the trends are similar across datasets, we decided to aggregate the results from all the datasets to enhance readability. We show the trends separately for each dataset in the Appendix section (Sect. 7).

[1] https://github.com/lurovi/cslim.

Table 2 Table with the median of RMSE on the test set and $\log_{10}(\ell)$ of the best individual for all the different methods grouped by exploration pipeline and algorithm

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	$\mathcal{E}_{\text{cx,mut}}$												
GP	$\mathcal{T}^0$	**22.67**$^+$	**9.54**$^+$	**5.28**$^+$	**10.41***	**2.8***	**1.45**$^+$	3.04	2.9	**2.55**	2.65	2.7	2.83
	$\mathcal{T}_2^2$	25.86	10.27$^+$	5.56	10.72	3.55	1.47	2.91	2.73$^+$	2.56	**2.28**$^+$	2.5	2.39$^+$
	$\mathcal{T}_3^2$	25.18	11.93	6.33	10.67	4.08	1.5	**2.78**	**2.66**$^+$	2.58	2.45$^+$	**2.47**	**2.33**$^+$
GSGP	$\mathcal{T}^0$	15.13	7.48	4.18	9.91	5.17	**1.23**	50.87	72.96	96.02	50.69	65.47	**16766***
	$\mathcal{T}_2^2$	14.82$^+$	**6.98**$^+$	**3.5**$^+$	9.75	4.34$^+$	1.25	**26.69***	**60.76***	**80.99***	27.29$^+$	**40.93**$^+$	190.1
	$\mathcal{T}_3^2$	**14.01***	6.99$^+$	3.66$^+$	**9.71**$^+$	**4.14***	1.27	27.71$^+$	67.57$^+$	91.7$^+$	**25.14**$^+$	42.9$^+$	185.4$^+$
$\text{SLIM}^+_{\text{ABS}}$	$\mathcal{T}^0$	12.39	8.19	3.84	10.01	4.31	1.31	**3.79***	**3.52***	**3.38***	**3.25***	**3.61***	**2.76***
	$\mathcal{T}_2^2$	9.22$^+$	7.41$^+$	3.81	**9.87**$^+$	**3.62**$^+$	1.31	3.97$^+$	3.77$^+$	3.66$^+$	3.59$^+$	3.84$^+$	3.36$^+$
	$\mathcal{T}_3^2$	**9.19**$^+$	**7.36**$^+$	**3.52**	9.87$^+$	3.77$^+$	**1.28**	4.06	3.84	3.69	3.64	3.9	3.39
$\text{SLIM}^+_{\text{SIG1}}$	$\mathcal{T}^0$	13.37	8.38	4.72	10.03	4.49	1.31	**3.76***	**3.47***	**3.3***	**3.42***	**3.53***	**2.81***
	$\mathcal{T}_2^2$	10.67$^+$	7.52$^+$	4.5	**9.9**$^+$	**3.45***	**1.3**	3.97$^+$	3.73$^+$	3.62$^+$	3.72$^+$	3.76$^+$	3.41$^+$
	$\mathcal{T}_3^2$	**9.88**$^+$	**7.45**$^+$	**4.25**	9.92$^+$	4.03$^+$	1.31	4.03	3.8	3.68	3.79	3.84	3.48
$\text{SLIM}^+_{\text{SIG2}}$	$\mathcal{T}^0$	14.07	8.36	4.81	10.03	3.96	1.28	**3.91***	**3.69***	**3.56***	**3.54***	**3.72***	**3.14***
	$\mathcal{T}_2^2$	12.04$^+$	7.52$^+$	4.4	9.85$^+$	3.63$^+$	1.28	4.13$^+$	3.94$^+$	3.83$^+$	3.82$^+$	3.95$^+$	3.63$^+$
	$\mathcal{T}_3^2$	**10.52**$^+$	**7.35**$^+$	**4.16**	**9.71**$^+$	**3.5**$^+$	**1.27**	4.21	4.01	3.88	3.9	4.04	3.69

For each group, we highlight in bold the variants with the lowest value (error or size). Moreover, we use a plus to highlight the methods that outperform at least another method in the same group and we use an asterisk to highlight the methods that outperform all the other methods in the same group

Table 2 reports the median performance and size aggregated over the algorithms (focusing on $\mathcal{E}_{\text{cx,mut}}$ pipeline), allowing us to study the influence of the cellular structure on both metrics.

For GP, which we recall is tested only with the $\mathcal{E}_{\text{cx,mut}}$ pipeline, better performance is generally obtained without a cellular structure, while the presence of the grid helps to reduce the size of the solutions.

Regarding GSGP, results with the $\mathcal{E}_{\text{cx}}$ pipeline are always poor (this result is only shown in the full Appendix table). In pipelines where mutation is present (alone or in combination with crossover), larger neighborhoods improve performance, while smaller neighborhoods are preferred for controlling solution size. The same behavior holds for all SLIM variants. This is expected: increasing the radius enlarges the portion of the population contributing to variation, allowing individuals to combine more diverse components. As a result, models grow faster, since semantic operators introduce larger modifications, often leading to more accurate but less compact solutions.

If mutation is the only operator employed, the non-cellular variants always lead to smaller models (this result is only shown in the full Appendix table).

Table 3 focuses on algorithmic comparisons. We begin with solution size. In pipelines where GP is present ($\mathcal{E}_{\text{cx,mut}}$), it produces smaller models than all other methods. However, while the gap with GSGP is at least one order of magnitude in $\log_{10}(\ell)$ (regardless of the topology), all SLIM variants achieve much more competitive sizes. When crossover is removed ($\mathcal{E}_{\text{mut}}$ pipeline), and GP is no longer present, SLIM consistently outperforms GSGP in model size, regardless of the cellular configuration. Among the SLIM variants, $\text{SLIM}^{+}_{\text{SIG1}}$ generally produces the smallest individuals. This aligns with the underlying operator design: this variant applies mutation using only one individual, generating smaller offspring.

Turning to performance, GSGP usually achieves the best results, with $\text{SLIM}^{+}_{\text{ABS}}$ performing comparably in some cases. The lowest accuracy is typically observed with GP.

Table 4 compares pipelines. As previously noted, GSGP with crossover only ($\mathcal{E}_{\text{cx}}$) performs worst across all datasets and topologies. This is consistent with theoretical findings: geometric crossover has limited exploratory power, as it cannot produce individuals outside the convex hull of the current population [87, 88]. If the global optimum lies outside this region, the search becomes trapped in local optima–a behavior clearly visible in Fig. 6–while solutions are smaller than those produced by the $\mathcal{E}_{\text{mut}}$ pipeline. This is expected: mutation in GSGP adds terms and causes exponential growth. The models in $\mathcal{E}_{\text{cx}}$ are smaller than those in $\mathcal{E}_{\text{cx,mut}}$, but only because crossover alone gets stuck early.

In this table, we omit the SLIM algorithms since the comparison between $\mathcal{E}_{\text{cx,mut}}$ and $\mathcal{E}_{\text{mut}}$, given the SLIM algorithm and the cellular topology, always leads to $\mathcal{E}_{\text{cx,mut}}$ being able to outperform $\mathcal{E}_{\text{mut}}$ as regards both error and size (this result is only shown in the full Appendix table).

For GSGP, the best pipeline depends on the cellular structure. As the neighborhood size increases, mutation-only pipelines tend to perform better, while crossover becomes less effective. This may be due to the broader range of individuals participating in variation, which reduces the need for crossover when the search already

Table 3 Table with the median of RMSE on the test set and $\log_{10}(\ell)$ of the best individual for all the different methods grouped by cellular structure and exploration pipeline

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	$\mathcal{T}^0$												
$\mathcal{E}_{\text{cx,mut}}$	GP	22.67	9.54	5.28	10.41	**2.8***	1.45	**3.04***	**2.9***	**2.55***	**2.65***	**2.7***	2.83^{+}
	GSGP	15.13^{+}	**7.48***	4.18^{+}	$\mathbf{9.91}^{+}$	5.17	**1.23***	50.87	72.96	96.02	50.69	65.47	167.66
	$\text{SLIM}^{+}_{\text{ABS}}$	$\mathbf{12.39}^{+}$	8.19^{+}	**3.84***	10.01^{+}	4.31^{+}	1.31^{+}	3.79^{+}	3.52^{+}	3.38^{+}	3.25^{+}	3.61^{+}	$\mathbf{2.76}^{+}$
$\mathcal{E}_{\text{mut}}$	GSGP	**15.2***	**7.59***	**3.88***	**9.92***	**4.84***	1.34	4.26	4.08	4.03	3.89	4.19	3.88
	$\text{SLIM}^{+}_{\text{ABS}}$	19.25	9.32	4.96	10.19	6.37^{+}	1.32	**3.74***	3.58^{+}	3.46^{+}	**3.29***	3.68^{+}	2.74^{+}
	$\text{SLIM}^{+}_{\text{SIG1}}$	17.38^{+}	8.69	4.61	10.22	6.82^{+}	1.32	3.76^{+}	**3.51***	**3.34***	3.43^{+}	**3.57***	$\mathbf{2.73}^{+}$
	$\mathcal{T}^2_2$												
$\mathcal{E}_{\text{cx,mut}}$	GP	25.86	10.27	5.56	10.72	3.55^{+}	1.47	**2.91***	**2.73***	**2.56***	**2.28***	**2.5***	**2.39***
	GSGP	14.82^{+}	**6.98***	$\mathbf{3.5}^{+}$	$\mathbf{9.75}^{+}$	4.34	$\mathbf{1.25}^{+}$	26.69	60.76	80.99	27.29	40.93	190.1
$\mathcal{E}_{\text{mut}}$	GSGP	**11.3***	**6.96***	$\mathbf{3.85}^{+}$	**9.75***	**3.97***	$\mathbf{1.27}^{+}$	4.32	4.13	4.06	3.94	4.24	4.03
	$\text{SLIM}^{+}_{\text{ABS}}$	15.75	8.13	3.86	10.08	5.23^{+}	1.29	4.05^{+}	3.83^{+}	3.7^{+}	**3.55***	3.93^{+}	**3.18***
	$\text{SLIM}^{+}_{\text{SIG1}}$	14.57^{+}	7.72	4.21	10.1	5.78^{+}	1.3	$\mathbf{4.04}^{+}$	**3.78***	**3.62***	3.77^{+}	**3.89***	3.27^{+}
	$\mathcal{T}^2_3$												
$\mathcal{E}_{\text{cx,mut}}$	GP	25.18	11.93	6.33	10.67	4.08	1.5	**2.78***	**2.66***	**2.58***	**2.45***	**2.47***	**2.33***
	GSGP	14.01^{+}	$\mathbf{6.99}^{+}$	3.66^{+}	$\mathbf{9.71}^{+}$	4.14	$\mathbf{1.27}^{+}$	27.71	67.57	91.7	25.14	42.9	185.4
	$\text{SLIM}^{+}_{\text{ABS}}$	$\mathbf{9.19}^{+}$	7.36^{+}	$\mathbf{3.52}^{+}$	9.87^{+}	3.77^{+}	1.28^{+}	4.06^{+}	3.84^{+}	3.69^{+}	3.64^{+}	3.9^{+}	3.39^{+}
$\mathcal{E}_{\text{mut}}$	GSGP	**10.15***	**6.83***	3.97^{+}	**9.69***	**3.74***	1.29	4.33	4.15	4.08	3.96^{+}	4.26	4.06
	$\text{SLIM}^{+}_{\text{ABS}}$	14.94	7.91	$\mathbf{3.87}^{+}$	10.04	4.89^{+}	1.29	4.13^{+}	3.91^{+}	3.78^{+}	**3.62***	4.0^{+}	**3.3***
	$\text{SLIM}^{+}_{\text{SIG1}}$	13.41^{+}	7.61	4.31	10.06	5.4^{+}	1.31	**4.12***	**3.87***	**3.73***	3.86^{+}	**3.97***	3.42^{+}

For each group, we highlight in bold the variants with the lowest value (error or size). Moreover, we use a plus to highlight the methods that outperform at least another method in the same group and we use an asterisk to highlight the methods that outperform all the other methods in the same group

spans distant regions of the grid. Moreover, with a larger neighborhood, crossover may involve more diverse individuals, increasing its disruptive potential and possibly leading to the loss of useful patterns.

Interestingly, in all SLIM variants, the $\mathcal{E}_{\mathrm{cx,mut}}$ pipeline outperforms others in both error and size, unlike in GSGP. This confirms the effectiveness of the proposed crossover operator in exchanging genetic material without compromising model compactness, while also improving search. As shown in Fig. 6, the improvement from crossover appears early and persists throughout evolution.

Figure 6 also highlights that the effect of crossover is stronger in standard SLIM (without a grid), suggesting that structured populations reduce the marginal benefit of recombination. Additionally, the same figure shows that GP generally stops improving after a few epochs, apart from minor exceptions. This is not the case for GSGP and SLIM, particularly on datasets such as YCH and QSR (as shown in the Appendix section), reinforcing the impact of the geometric properties that differentiate them from standard GP.

Finally, Fig. 7 confirms that the trends observed for solution size in the tables hold consistently at each generation.

5.2 Diversity

Following the intuition of [7], we use Moran's *I* to analyze diversity in cellular methods. In Fig. 8, we track the trend of *I* over generations, aggregating values from all datasets and repetitions (the plot showing the trend for each dataset separately is available in the Appendix section).

We recall that the term $\mathcal{T}^0$ refers to the non-cellular standard version, while $\mathcal{T}_2^2$ and $\mathcal{T}_3^2$ indicate, respectively, cellular methods with radius 2 and 3. Moreover, we use $\mathcal{E}_{\mathrm{cx}}$ to denote methods that only perform crossover, $\mathcal{E}_{\mathrm{mut}}$ for methods that only perform mutation, and $\mathcal{E}_{\mathrm{cx,mut}}$ for methods performing both crossover and mutation in a mutually exclusive way for each individual.

Given the consistency of results across pipelines, we direct our discussion toward the effects of algorithms and cellular structures. For GSGP and SLIM^{+}, the grid preserves diversity in the early stages of evolution, slowing the spread of dominant individuals by confining them within local neighborhoods. Over time, these solutions propagate across the grid and *I* converges to zero.

This confirms that cellular methods outperform their non-cellular counterparts when semantic diversity is preserved for part of the search. Conversely, when *I* remains low throughout evolution, as in GP, the grid does not provide any benefit, since individuals remain randomly distributed.

The plots also show that, for GP, good and poor individuals are often placed close together, with no clear spatial organization. In contrast, for GSGP, dominant solutions spread gradually, allowing the grid to maintain spatial clusters for longer and guide the search more effectively.

This behavior reflects the nature of the operators: additive transformations preserve local patterns, while disruptive operators quickly erase them.

Additive transformations are genetic operators that modify individuals by adding new components to their structure using mathematical addition. In GP, these opera-

Table 4 Table with the median of RMSE on the test set and $\log_{10}(\ell)$ of the best individual for all the different methods grouped by algorithm and cellular structure

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	GSGP												
$\mathcal{T}^0$	$\mathcal{E}_{\text{cx,mut}}$	$\mathbf{15.13}^+$	$\mathbf{7.48}^+$	4.18^+	$\mathbf{9.91}^+$	5.17^+	**1.23***	50.87	72.96	96.02	50.69	65.47	167.66
	$\mathcal{E}_{\text{cx}}$	64.11	20.94	10.34	12.43	14.24	1.74	6.75^+	14.26^+	11.4^+	9.22^+	10.25^+	17.23^+
	$\mathcal{E}_{\text{mut}}$	15.2^+	7.59^+	$\mathbf{3.88}^+$	9.92^+	**4.84***	1.34^+	**4.26***	**4.08***	**4.03***	**3.89***	**4.19***	**3.88***
$\mathcal{T}_2^2$	$\mathcal{E}_{\text{cx,mut}}$	14.82^+	6.98^+	$\mathbf{3.5}^+$	9.75^+	4.34^+	$\mathbf{1.25}^+$	26.69	60.76	80.99	27.29	40.93	190.1
	$\mathcal{E}_{\text{cx}}$	63.29	29.62	11.84	12.99	14.41	1.81	**3.35***	$\mathbf{3.71}^+$	4.57^+	$\mathbf{3.9}^+$	4.6^+	7.48^+
	$\mathcal{E}_{\text{mut}}$	**11.3***	$\mathbf{6.96}^+$	3.85^+	$\mathbf{9.75}^+$	$\mathbf{3.97}^+$	1.27^+	4.32^+	4.13^+	**4.06***	3.94^+	$\mathbf{4.24}^+$	**4.03***
$\mathcal{T}_3^2$	$\mathcal{E}_{\text{cx,mut}}$	14.01^+	6.99^+	$\mathbf{3.66}^+$	9.71^+	4.14^+	$\mathbf{1.27}^+$	27.71	67.57	91.7	25.14	42.9	185.4
	$\mathcal{E}_{\text{cx}}$	65.11	29.83	11.44	13.26	14.42	1.82	**2.36***	**2.89***	4.37^+	**3.45***	$\mathbf{4.05}^+$	5.51^+
	$\mathcal{E}_{\text{mut}}$	**10.15***	$\mathbf{6.83}^+$	3.97^+	$\mathbf{9.69}^+$	$\mathbf{3.74}^+$	1.29^+	4.33^+	4.15^+	$\mathbf{4.08}^+$	3.96^+	4.26^+	**4.06***

For each group, we highlight in bold the variants with the lowest value (error or size). Moreover, we use a plus to highlight the methods that outperform at least another method in the same group and we use an asterisk to highlight the methods that outperform all the other methods in the same group

tors (e.g., those used in GSGP and SLIM^+) typically combine parent individuals in a way that preserves their core semantics and structural patterns, resulting in offspring that retain much of their parents' behavior.

Disruptive operators, on the other hand, are genetic operators that produce offspring that are structurally or behaviorally very dissimilar from their parents. These operators, such as standard sub-tree crossover, often replace large parts of a parent solution, which can erase previously learned patterns and introduce significant variation–potentially both beneficial and detrimental–into the population.

In cellular structures, this balance between preserving diversity within neighborhoods and controlling information diffusion is key to achieving better solutions.

5.3 Algorithms Pareto front

We summarize our main findings and provide a global comparison of all algorithms through Pareto fronts that highlight the trade-offs between model error and model size (Fig. 9).

We recall that the term $\mathcal{T}^0$ refers to the non-cellular standard version, while $\mathcal{T}_2^2$ and $\mathcal{T}_3^2$ indicate, respectively, cellular methods with radius 2 and 3. Moreover, we use $\mathcal{E}_{\text{cx}}$ to denote methods that only perform crossover, $\mathcal{E}_{\text{mut}}$ for methods that only perform mutation, and $\mathcal{E}_{\text{cx,mut}}$ for methods performing both crossover and mutation in a mutually exclusive way for each individual.

To better visualize the behavior of the different methods – since some GSGP-based variants exhibit higher errors that compress the other results – we also report a zoomed version in Fig. 10. Specifically, we removed from this plot the algorithms leading to the worst solutions, namely GSGP with the $\mathcal{E}_{\text{cx}}$ pipeline (regardless of the cellular structure), and the algorithms producing the largest individuals, namely GSGP with the $\mathcal{E}_{\text{cx,mut}}$ pipeline, where the presence of GSC causes exponential growth in solution size. We additionally removed the GP variants to enhance readability (in the Appendix section, a zoomed version of the original plot can be found including also GP and crossover-only GSGP).

Results are aggregated across all datasets and repetitions for each algorithm and method.

The figure shows that, in terms of performance, two methods stand out: $\mathcal{E}_{\text{mut}}$-GSGP and $\mathcal{E}_{\text{cx,mut}}$-$\text{SLIM}^+_{\text{ABS}}$, with the latter also producing smaller models. Removing the grid in $\mathcal{E}_{\text{cx,mut}}$-$\text{SLIM}^+_{\text{ABS}}$ results in smaller individuals, but at the expense of performance. This behavior is commonly observed: the best results are often obtained in the presence of a grid, at the cost of an increase in model size.

Another consistent pattern is that all SLIM variants benefit from the addition of crossover in terms of performance, while the increase in size remains very limited. Thus, we can conclude that introducing crossover in SLIM is a good design choice.

As noted in the previous analysis, GP produces the smallest solutions and is the only algorithm for which the cellular variant results in smaller models than its non-cellular counterpart–albeit with a reduction in performance. In general, SLIM^+ achieves more accurate models when combined with a cellular structure.

These results highlight the trade-offs between error and model size when selecting a GP algorithm for SR. Typically, GSGP offers the best performance, while GP

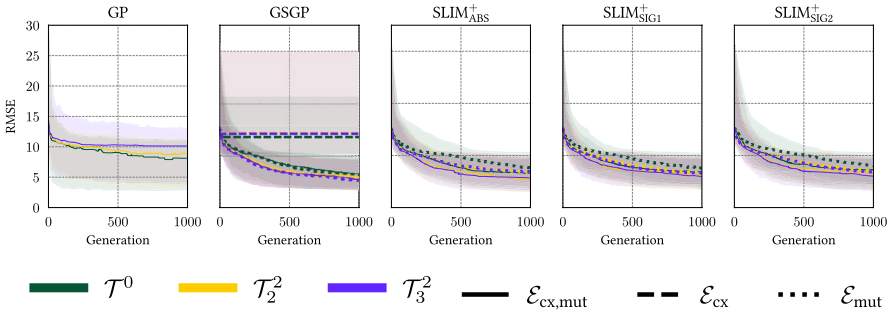

Fig. 6 Evolution of test fitness for the best individual across 30 runs

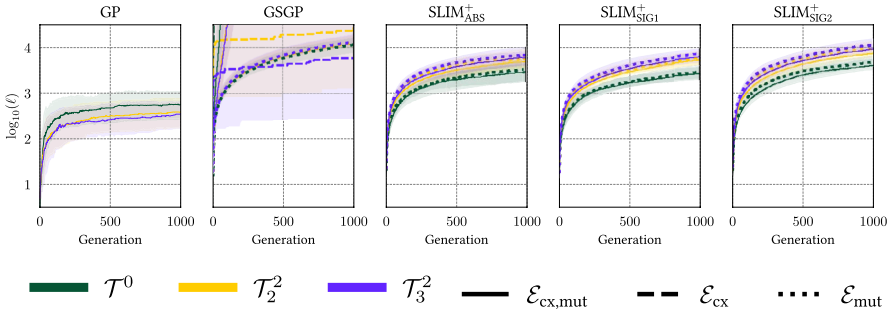

Fig. 7 Evolution of the size of the best individual across 30 runs

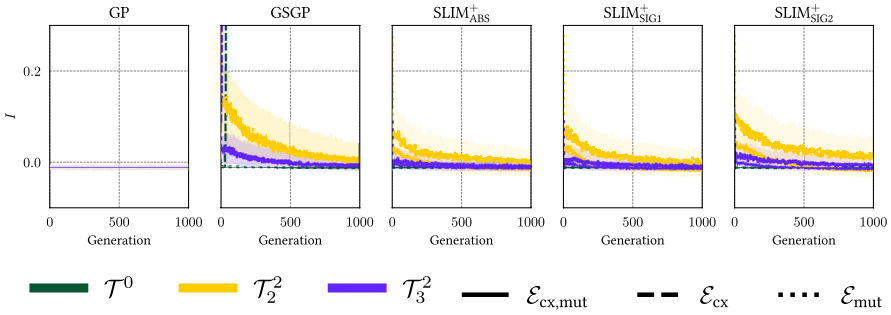

Fig. 8 Trend of the Global Moran's I calculated on the population

ensures the highest interpretability. Cellular SLIM variants lie in between, providing varying levels of compromise between these two aspects.

As a general rule of thumb, cellular structures are worth employing when using algorithms based on non-disruptive operators, where new solutions remain similar to their parents.

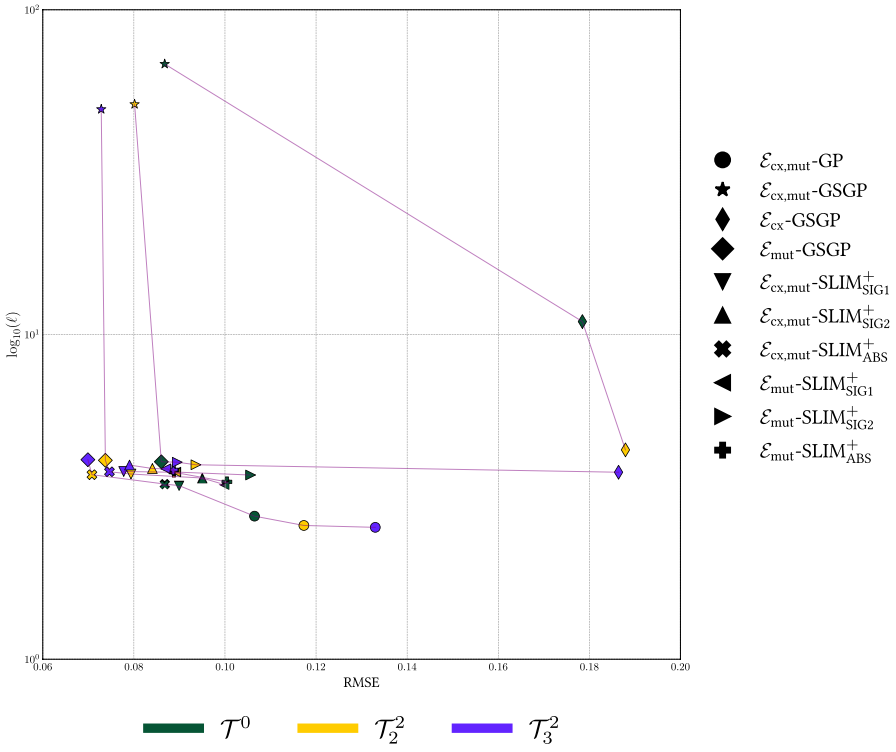

Fig. 9 Pareto fronts of the tested methods. Each Pareto front represents a set of non-dominated solutions w.r.t. to both error and size. Each method is identified by the median across all datasets and repetitions. RMSE is scaled in [0, 1] by using the maximum RMSE discovered among the best solutions from all the experiments

5.4 Discussion

We now provide explicit answers to the research questions introduced at the beginning of Sect. 4.

RQ1 *How does the introduction of a cellular (toroidal) structure affect the performance and size of evolved solutions in different GP algorithms?* The impact of cellular structures depends strongly on the algorithm. For standard GP, the toroidal structure helps reduce solution size but slightly harms performance. In contrast, for semantic algorithms such as GSGP and SLIM, the grid generally improves predictive performance, particularly when additive operators are used. This is due to the preservation of semantic diversity in early generations, as confirmed by the behavior of Moran's I (Fig. 8). However, the introduction of a grid usually results in larger models.

RQ2 *How do different exploration pipelines (i.e., combinations of mutation and crossover) influence accuracy and model size, especially in semantic methods such as GSGP and SLIM?* In GSGP, pipelines that rely only on crossover ($\mathcal{E}_{cx}$) perform poorly, both in terms of accuracy and diversity, due to the limited exploratory power of geometric crossover. Pipelines based sole-

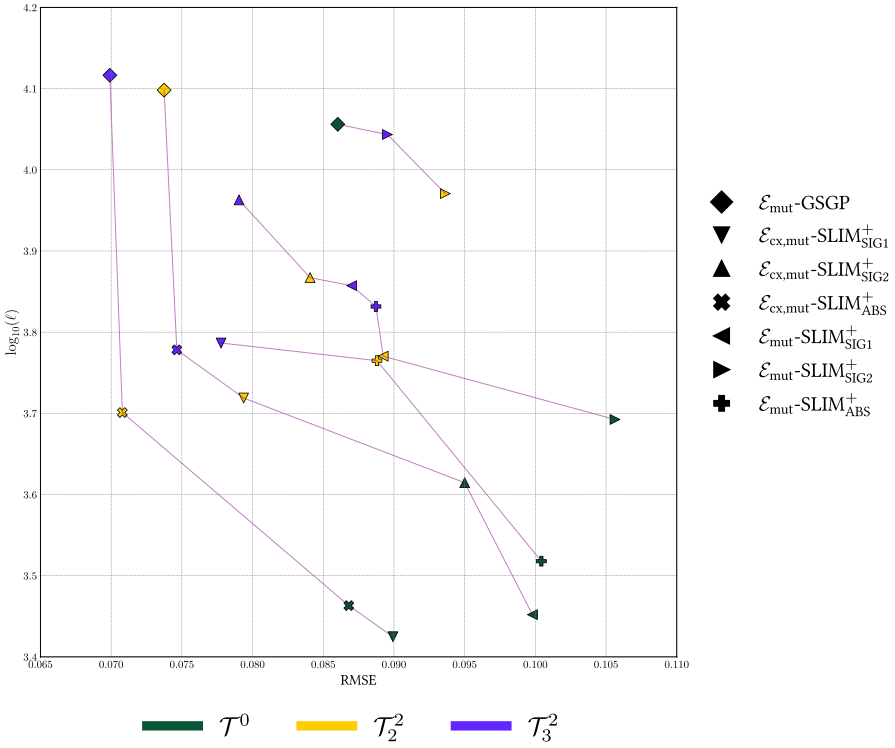

Fig. 10 Zoomed version of Fig. 9

ly on mutation ($\mathcal{E}_{\text{mut}}$) tend to yield the best performance, although they also lead to substantial growth in model size. Mixed pipelines ($\mathcal{E}_{\text{cx,mut}}$) strike a balance but typically produce larger models than mutation-only variants. For SLIM, the proposed crossover operator proves highly effective: the $\mathcal{E}_{\text{cx,mut}}$ pipeline consistently outperforms mutation-only in both error and size, confirming that SLIM benefits significantly from recombination.

RQ3 *Which combinations of algorithm, exploration pipeline, and spatial structure achieve the best trade-off between accuracy and interpretability (measured by model size)?* The best trade-off is achieved by SLIM$^{+}_{\text{ABS}}$ with the $\mathcal{E}_{\text{cx,mut}}$ pipeline and a toroidal grid of radius 2, which combines high predictive accuracy with competitive model size. In general, GP produces the most compact models. GSGP achieves the best accuracy, especially with the $\mathcal{E}_{\text{mut}}$ pipeline, but suffers from exponential growth in model size. Cellular SLIM variants provide several levels of trade-off.

RQ4 *To what extent do cellular structures promote semantic diversity during evolution, and how does this affect search dynamics?* As shown by the evolution of Global Moran's I (Fig. 8), cellular structures effectively maintain semantic diversity in the early stages of evolution in GSGP and SLIM^{+}. The grid slows the diffusion of dominant solutions, encouraging parallel exploration in different regions of the search space. This effect is beneficial when operators are additive and preserve se-

mantic patterns. In contrast, when disruptive operators are used (e.g., in GP), the grid does not produce significant spatial organization, and spatial pattern quickly decays. These observations confirm that spatial structures enhance search effectiveness mainly in the presence of smooth semantic variation.

6 Conclusion and future work

In this paper, we investigated the impact of incorporating CA-inspired spatial structures into three GP approaches: standard GP, GSGP, and SLIM. This work builds on our previous study that first introduced a cellular structure in SLIM [26], focusing solely on semantic mutation. Here, we extended that framework by incorporating semantic crossover and analyzing its interaction with spatial constraints.

Experiments on six symbolic regression problems show that cellular structures improve the search dynamics in GSGP and SLIM^{+} (the SLIM variant that employs additive mutation), particularly when genetic operators produce smooth, additive semantic changes. These improvements result in higher predictive performance and greater semantic diversity in the early stages of evolution. Conversely, when operators are more disruptive, such as standard crossover in GP, the benefit of the grid is limited, and spatial patterns are quickly lost.

The analysis based on Global Moran's I confirms that structured populations are most effective when diversity is preserved within neighborhoods for a sufficient part of the search. This explains why the grid consistently improves performance in GSGP and SLIM^{+}, while GP mainly benefits in terms of solution size reduction.

Overall, our results highlight different trade-offs that are valid across the examined methods: GP remains a good choice for producing compact models, GSGP provides the highest accuracy, and SLIM offers a flexible compromise between these two objectives. In particular, the proposed crossover operator for SLIM proved effective, improving performance with a very limited impact on model size.

Future work will extend this framework to other semantic GP variants, explore adaptive neighborhood schemes, and assess robustness under noisy or evolving data. Integrating multi-objective formulations may also offer further insights into balancing model size and accuracy within spatially structured semantic GP.

Appendix

In the Appendix section, we report the tables and plots containing the complete results of our experimental campaign (Figs. 11, 12, 13 and 14), (Tables 5, 6 and 7).

Table 5 Table with the median of RMSE on the test set and $\log_{10}(\ell)$ of the best individual for all the different methods grouped by exploration pipeline and algorithm

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	$\mathcal{E}_{\text{cx,mut}}$												
GP	$\mathcal{T}^0$	**22.67**$^+$	**9.54**$^+$	**5.28**$^+$	**10.41***	**2.8***	**1.45**$^+$	3.04	2.9	**2.55**	2.65	2.7	2.83
	$\mathcal{T}_2^2$	25.86	10.27$^+$	5.56	10.72	3.55	1.47	2.91	2.73$^+$	2.56	**2.28**$^+$	2.5	2.39$^+$
	$\mathcal{T}_3^2$	25.18	11.93	6.33	10.67	4.08	1.5	**2.78**	**2.66**$^+$	2.58	2.45$^+$	**2.47**	**2.33**$^+$
GSGP	$\mathcal{T}^0$	15.13	7.48	4.18	9.91	5.17	**1.23**	50.87	72.96	96.02	50.69	65.47	**16766***
	$\mathcal{T}_2^2$	14.82$^+$	**6.98**$^+$	**3.5**$^+$	9.75	4.34$^+$	1.25	**26.69***	**60.76***	**80.99***	27.29$^+$	**40.93**$^+$	190.1
	$\mathcal{T}_3^2$	**14.01***	6.99$^+$	3.66$^+$	**9.71**$^+$	**4.14***	1.27	27.71$^+$	67.57$^+$	91.7$^+$	**25.14**$^+$	42.9$^+$	185.4$^+$
$\text{SLIM}^+_{\text{ABS}}$	$\mathcal{T}^0$	12.39	8.19	3.84	10.01	4.31	1.31	**3.79***	**3.52***	**3.38***	**3.25***	**3.61***	**2.76***
	$\mathcal{T}_2^2$	9.22$^+$	7.41$^+$	3.81	**9.87**$^+$	**3.62**$^+$	1.31	3.97$^+$	3.77$^+$	3.66$^+$	3.59$^+$	3.84$^+$	3.36$^+$
	$\mathcal{T}_3^2$	**9.19**$^+$	**7.36**$^+$	**3.52**	9.87$^+$	3.77$^+$	**1.28**	4.06	3.84	3.69	3.64	3.9	3.39
$\text{SLIM}^+_{\text{SIG1}}$	$\mathcal{T}^0$	13.37	8.38	4.72	10.03	4.49	1.31	**3.76***	**3.47***	**3.3***	**3.42***	**3.53***	**2.81***
	$\mathcal{T}_2^2$	10.67$^+$	7.52$^+$	4.5	**9.9**$^+$	**3.45***	**1.3**	3.97$^+$	3.73$^+$	3.62$^+$	3.72$^+$	3.76$^+$	3.41$^+$
	$\mathcal{T}_3^2$	**9.88**$^+$	**7.45**$^+$	**4.25**	9.92$^+$	4.03$^+$	1.31	4.03	3.8	3.68	3.79	3.84	3.48
$\text{SLIM}^+_{\text{SIG2}}$	$\mathcal{T}^0$	14.07	8.36	4.81	10.03	3.96	1.28	**3.91***	**3.69***	**3.56***	**3.54***	**3.72***	**3.14***
	$\mathcal{T}_2^2$	12.04$^+$	7.52$^+$	4.4	9.85$^+$	3.63$^+$	1.28	4.13$^+$	3.94$^+$	3.83$^+$	3.82$^+$	3.95$^+$	3.63$^+$
	$\mathcal{T}_3^2$	**10.52**$^+$	**7.35**$^+$	**4.16**	**9.71**$^+$	**3.5**$^+$	**1.27**	4.21	4.01	3.88	3.9	4.04	3.69
	$\mathcal{E}_{\text{cx}}$												
GSGP	$\mathcal{T}^0$	64.11	**20.94***	**10.34**$^+$	**12.43**	**14.24**	**1.74**	6.75	14.26	11.4	9.22	10.25	17.23
	$\mathcal{T}_2^2$	**63.29**	29.62	11.84	12.99	14.41	1.81	3.35$^+$	3.71$^+$	4.57$^+$	3.9$^+$	4.6$^+$	7.48$^+$
	$\mathcal{T}_3^2$	65.11	29.83	11.44	13.26	14.42	1.82	**2.36**$^+$	**2.89***	**4.37**$^+$	**3.45**$^+$	**4.05**$^+$	**5.51***
	$\mathcal{E}_{\text{mut}}$												
GSGP	$\mathcal{T}^0$	15.2	7.59	3.88	9.92	4.84	1.34	**4.26***	**4.08***	**4.03***	**3.89***	**4.19***	**3.88***
	$\mathcal{T}_2^2$	11.3$^+$	6.96$^+$	**3.85**	9.75$^+$	3.97$^+$	**1.27**$^+$	4.32$^+$	4.13$^+$	4.06$^+$	3.94$^+$	4.24$^+$	4.03$^+$
	$\mathcal{T}_3^2$	**10.15***	**6.83**$^+$	3.97	**9.69**$^+$	**3.74**$^+$	1.29$^+$	4.33	4.15	4.08	3.96	4.26	4.06

Table 5 (continued)

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	$\mathcal{E}_{\text{cx,mut}}$												
$\text{SLIM}^{+}_{\text{ABS}}$	$\mathcal{T}^0$	19.25	9.32	4.96	10.19	6.37	1.32	**3.74***	**3.58***	**3.46***	**3.29***	**3.68***	**2.74***
	$\mathcal{T}^2_2$	15.75^{+}	8.13^{+}	$\mathbf{3.86}^{+}$	10.08^{+}	5.23^{+}	1.29	4.05^{+}	3.83^{+}	3.7^{+}	3.55^{+}	3.93^{+}	3.18^{+}
	$\mathcal{T}^2_3$	**14.94***	$\mathbf{7.91}^{+}$	3.87^{+}	$\mathbf{10.04}^{+}$	**4.89***	**1.29**	4.13	3.91	3.78	3.62	4.0	3.3
$\text{SLIM}^{+}_{\text{SIG1}}$	$\mathcal{T}^0$	17.38	8.69	4.61	10.22	6.82	1.32	**3.76***	**3.51***	**3.34***	**3.43***	**3.57***	**2.73***
	$\mathcal{T}^2_2$	14.57^{+}	7.72^{+}	**4.21**	10.1^{+}	5.78^{+}	**1.3**	4.04^{+}	3.78^{+}	3.62^{+}	3.77^{+}	3.89^{+}	3.27^{+}
	$\mathcal{T}^2_3$	**13.41***	$\mathbf{7.61}^{+}$	4.31	$\mathbf{10.06}^{+}$	**5.4***	1.31	4.12	3.87	3.73	3.86	3.97	3.42
$\text{SLIM}^{+}_{\text{SIG2}}$	$\mathcal{T}^0$	19.74	9.15	4.59	10.23	7.34	**1.27**	**3.94***	**3.75***	**3.63***	**3.6***	**3.79***	**3.13***
	$\mathcal{T}^2_2$	15.53^{+}	7.66^{+}	4.32	9.98^{+}	6.22^{+}	1.28	4.24^{+}	4.02^{+}	3.9^{+}	3.91^{+}	4.11^{+}	3.57^{+}
	$\mathcal{T}^2_3$	**14.53***	$\mathbf{7.58}^{+}$	**4.22**	$\mathbf{9.93}^{+}$	**5.9***	1.28	4.3	4.09	3.96	3.99	4.19	3.69

For each group, we highlight in bold the variants with the lowest value (error or size). Moreover, we use a plus to highlight the methods that outperform at least another method in the same group and we use an asterisk to highlight the methods that outperform all the other methods in the same group

Table 6 Table with the median of RMSE on the test set and $\log_{10}(\ell)$ of the best individual for all the different methods grouped by cellular structure and exploration pipeline

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	$\mathcal{T}^0$												
$\mathcal{E}_{\text{cx,mut}}$	GP	22.67	9.54	5.28	10.41	**2.8***	1.45	**3.04***	**2.9***	**2.55***	**2.65***	**2.7***	2.83^{+}
	GSGP	15.13^{+}	**7.48***	4.18^{+}	$\mathbf{9.91}^{+}$	5.17	**1.23***	50.87	72.96	96.02	50.69	65.47	167.66
	$\text{SLIM}^{+}_{\text{ABS}}$	$\mathbf{12.39}^{+}$	8.19^{+}	**3.84***	10.01^{+}	4.31^{+}	1.31^{+}	3.79^{+}	3.52^{+}	3.38^{+}	3.25^{+}	3.61^{+}	$\mathbf{2.76}^{+}$
	$\text{SLIM}^{+}_{\text{SIG1}}$	13.37^{+}	8.38^{+}	4.72	10.03^{+}	4.49^{+}	1.31^{+}	3.76^{+}	3.47^{+}	3.3^{+}	3.42^{+}	3.53^{+}	2.81^{+}
	$\text{SLIM}^{+}_{\text{SIG2}}$	14.07^{+}	8.36^{+}	4.81	10.03^{+}	3.96^{+}	1.28^{+}	3.91^{+}	3.69^{+}	3.56^{+}	3.54^{+}	3.72^{+}	3.14^{+}
$\mathcal{E}_{\text{mut}}$	GSGP	**15.2***	**7.59***	**3.88***	**9.92***	**4.84***	1.34	4.26	4.08	4.03	3.89	4.19	3.88
	$\text{SLIM}^{+}_{\text{ABS}}$	19.25	9.32	4.96	10.19	6.37^{+}	1.32	**3.74***	3.58^{+}	3.46^{+}	**3.29***	3.68^{+}	2.74^{+}
	$\text{SLIM}^{+}_{\text{SIG1}}$	17.38^{+}	8.69	4.61	10.22	6.82^{+}	1.32	3.76^{+}	**3.51***	**3.34***	3.43^{+}	**3.57***	$\mathbf{2.73}^{+}$
	$\text{SLIM}^{+}_{\text{SIG2}}$	19.74	9.15	4.59	10.23	7.34	**1.27***	3.94^{+}	3.75^{+}	3.63^{+}	3.6^{+}	3.79^{+}	3.13^{+}
	$\mathcal{T}^2_2$												
$\mathcal{E}_{\text{cx,mut}}$	GP	25.86	10.27	5.56	10.72	3.55^{+}	1.47	**2.91***	**2.73***	**2.56***	**2.28***	**2.5***	**2.39***
	GSGP	14.82^{+}	**6.98***	$\mathbf{3.5}^{+}$	$\mathbf{9.75}^{+}$	4.34	$\mathbf{1.25}^{+}$	26.69	60.76	80.99	27.29	40.93	190.1
	$\text{SLIM}^{+}_{\text{ABS}}$	$\mathbf{9.22}^{+}$	7.41^{+}	3.81^{+}	9.87^{+}	3.62^{+}	1.31^{+}	3.97^{+}	3.77^{+}	3.66^{+}	3.59^{+}	3.84^{+}	3.36^{+}
	$\text{SLIM}^{+}_{\text{SIG1}}$	10.67^{+}	7.52^{+}	4.5^{+}	9.9^{+}	$\mathbf{3.45}^{+}$	1.3^{+}	3.97^{+}	3.73^{+}	3.62^{+}	3.72^{+}	3.76^{+}	3.41^{+}
	$\text{SLIM}^{+}_{\text{SIG2}}$	12.04^{+}	7.52^{+}	4.4^{+}	9.85^{+}	3.63^{+}	1.28^{+}	4.13^{+}	3.94^{+}	3.83^{+}	3.82^{+}	3.95^{+}	3.63^{+}
$\mathcal{E}_{\text{mut}}$	GSGP	**10.15***	**6.83***	3.97^{+}	**9.69***	**3.74***	1.29	4.33	4.15	4.08	3.96^{+}	4.26	4.06
	$\text{SLIM}^{+}_{\text{ABS}}$	14.94	7.91	$\mathbf{3.87}^{+}$	10.04	4.89^{+}	1.29	4.13^{+}	3.91^{+}	3.78^{+}	**3.62***	4.0^{+}	**3.3***
	$\text{SLIM}^{+}_{\text{SIG1}}$	13.41^{+}	7.61	4.31	10.06	5.4^{+}	1.31	**4.12***	**3.87***	**3.73***	3.86^{+}	**3.97***	3.42^{+}
	$\text{SLIM}^{+}_{\text{SIG2}}$	14.53	7.58	4.22	9.93	5.9	$\mathbf{1.28}^{+}$	4.3^{+}	4.09^{+}	3.96^{+}	3.99	4.19^{+}	3.69^{+}

For each group, we highlight in bold the variants with the lowest value (error or size). Moreover, we use a plus to highlight the methods that outperform at least another method in the same group and we use an asterisk to highlight the methods that outperform all the other methods in the same group

Table 7 Table with the median of RMSE on the test set and $\log_{10}(\ell)$ of the best individual for all the different methods grouped by algorithm and cellular structure

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	GSGP												
$\mathcal{T}^0$	$\mathcal{E}_{\text{cx,mut}}$	**15.13**$^+$	**7.48**$^+$	4.18$^+$	**9.91**$^+$	5.17$^+$	**1.23***	50.87	72.96	96.02	50.69	65.47	167.66
	$\mathcal{E}_{\text{cx}}$	64.11	20.94	10.34	12.43	14.24	1.74	6.75$^+$	14.26$^+$	11.4$^+$	9.22$^+$	10.25$^+$	17.23$^+$
	$\mathcal{E}_{\text{mut}}$	15.2$^+$	7.59$^+$	**3.88**$^+$	9.92$^+$	**4.84***	1.34$^+$	**4.26***	**4.08***	**4.03***	**3.89***	**4.19***	**3.88***
$\mathcal{T}_2^2$	$\mathcal{E}_{\text{cx,mut}}$	14.82$^+$	6.98$^+$	**3.5**$^+$	9.75$^+$	4.34$^+$	**1.25**$^+$	26.69	60.76	80.99	27.29	40.93	190.1
	$\mathcal{E}_{\text{cx}}$	63.29	29.62	11.84	12.99	14.41	1.81	**3.35***	**3.71**$^+$	4.57$^+$	**3.9**$^+$	4.6$^+$	7.48$^+$
	$\mathcal{E}_{\text{mut}}$	**11.3***	**6.96**$^+$	3.85$^+$	**9.75**$^+$	**3.97**$^+$	1.27$^+$	4.32$^+$	4.13$^+$	**4.06***	3.94$^+$	**4.24**$^+$	**4.03***
$\mathcal{T}_3^2$	$\mathcal{E}_{\text{cx,mut}}$	14.01$^+$	6.99$^+$	**3.66**$^+$	9.71$^+$	4.14$^+$	**1.27**$^+$	27.71	67.57	91.7	25.14	42.9	185.4
	$\mathcal{E}_{\text{cx}}$	65.11	29.83	11.44	13.26	14.42	1.82	**2.36***	**2.89***	4.37$^+$	**3.45***	**4.05**$^+$	5.51$^+$
	$\mathcal{E}_{\text{mut}}$	**10.15***	**6.83**$^+$	3.97$^+$	**9.69**$^+$	**3.74**$^+$	1.29$^+$	4.33$^+$	4.15$^+$	**4.08**$^+$	3.96$^+$	4.26$^+$	**4.06***
	$\text{SLIM}^+_{\text{ABS}}$												
$\mathcal{T}^0$	$\mathcal{E}_{\text{cx,mut}}$	**12.39***	**8.19***	**3.84***	**10.01***	**4.31***	**1.31**	3.79	**3.52***	**3.38***	**3.25***	**3.61***	2.76
	$\mathcal{E}_{\text{mut}}$	19.25	9.32	4.96	10.19	6.37	1.32	**3.74***	3.58	3.46	3.29	3.68	**2.74***
$\mathcal{T}_2^2$	$\mathcal{E}_{\text{cx,mut}}$	**9.22***	**7.41***	**3.81**	**9.87***	**3.62***	1.31	**3.97***	**3.77***	**3.66***	3.59	**3.84***	3.36
	$\mathcal{E}_{\text{mut}}$	15.75	8.13	3.86	10.08	5.23	**1.29**	4.05	3.83	3.7	**3.55***	3.93	**3.18***
$\mathcal{T}_3^2$	$\mathcal{E}_{\text{cx,mut}}$	**9.88***	**7.45**	**4.25**	**9.92***	**4.03***	**1.27**	**4.03***	**3.8***	**3.68***	**3.79***	**3.84***	3.48
	$\mathcal{E}_{\text{mut}}$	13.41	7.61	4.31	10.06	5.4	**1.31**	4.12	3.87	3.73	3.86	3.97	**3.42***
	$\text{SLIM}^+_{\text{SIG1}}$												
$\mathcal{T}^0$	$\mathcal{E}_{\text{cx,mut}}$	**13.37***	**8.38**	4.72	**10.03***	**4.49***	**1.31**	**3.76**	**3.47***	**3.3***	**3.42**	**3.53***	2.81
	$\mathcal{E}_{\text{mut}}$	17.38	8.69	**4.61**	10.22	6.82	1.32	3.76	3.51	3.34	3.43	3.57	**2.73***
$\mathcal{T}_2^2$	$\mathcal{E}_{\text{cx,mut}}$	**10.67***	**7.52**	4.5	**9.9***	**3.45***	1.3	**3.97***	**3.73***	**3.62**	**3.72***	**3.76***	3.41
	$\mathcal{E}_{\text{mut}}$	14.57	7.72	**4.21**	10.1	5.78	**1.3**	4.04	3.78	3.62	3.77	3.89	**3.27***

Table 7 (continued)

		RMSE						$\log_{10}(\ell)$					
		ARF	CNC	SLM	YCH	PRK	QSR	ARF	CNC	SLM	YCH	PRK	QSR
	GSGP												
$\mathcal{T}_3^2$	$\mathcal{E}_{\mathrm{cx,mut}}$	**10.52***	**7.35**	**4.16**	**9.71***	**3.5***	**1.27**	**4.21***	**4.01***	**3.88***	**3.9***	**4.04***	**3.69**
	$\mathcal{E}_{\mathrm{mut}}$	14.53	7.58	4.22	9.93	5.9	1.28	4.3	4.09	3.96	3.99	4.19	3.69

For each group, we highlight in bold the variants with the lowest value (error or size). Moreover, we use a plus to highlight the methods that outperform at least another method in the same group and we use an asterisk to highlight the methods that outperform all the other methods in the same group

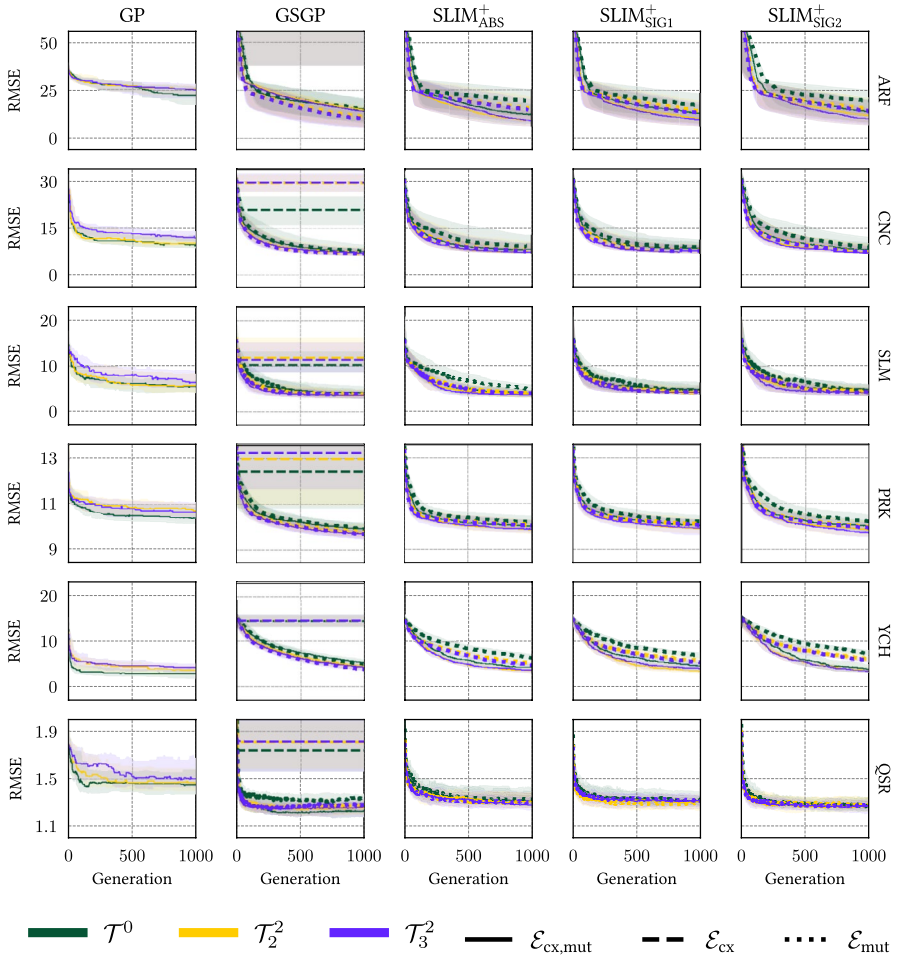

Fig. 11 Evolution of test fitness for the best individual across 30 runs

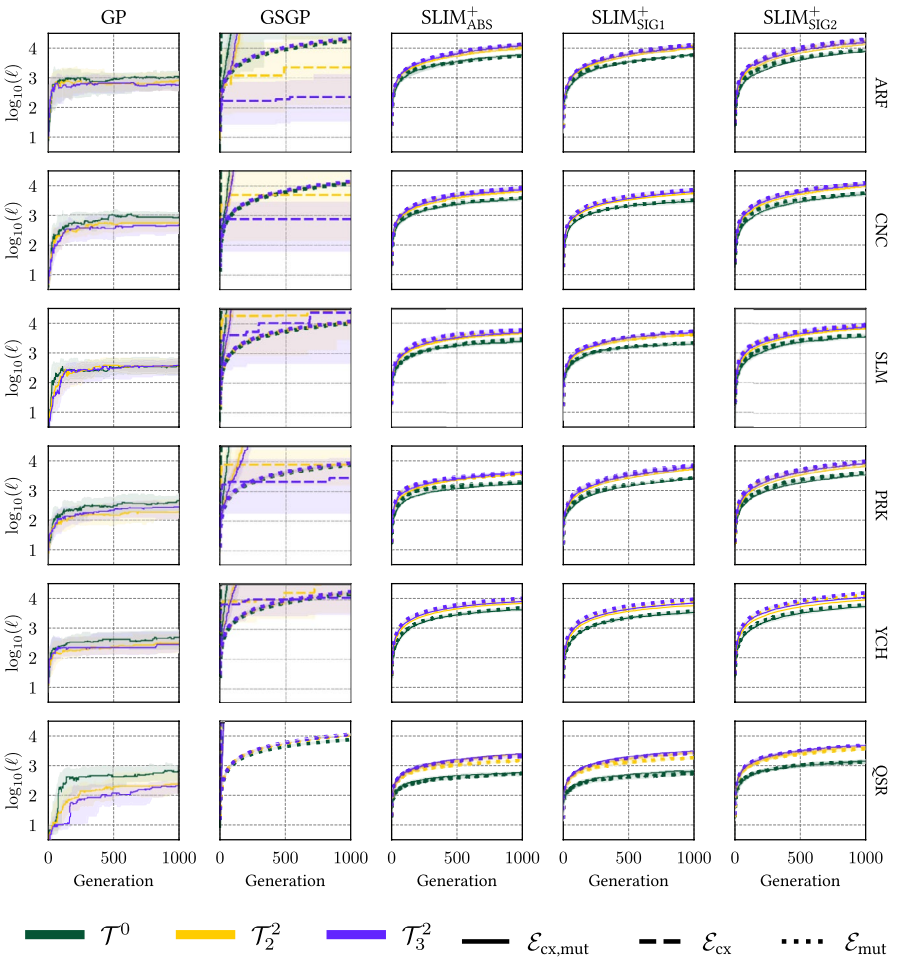

Fig. 12 Evolution of the size of the best individual across 30 runs

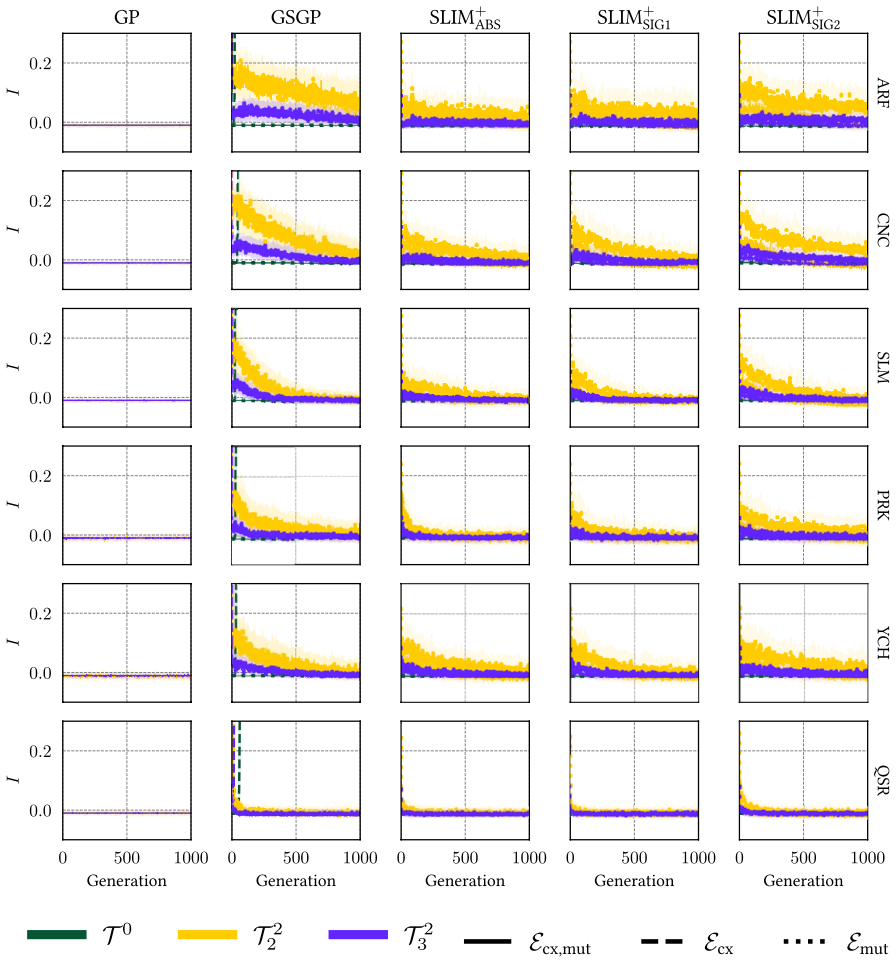

Fig. 13 Trend of the Global Moran's I calculated on the population

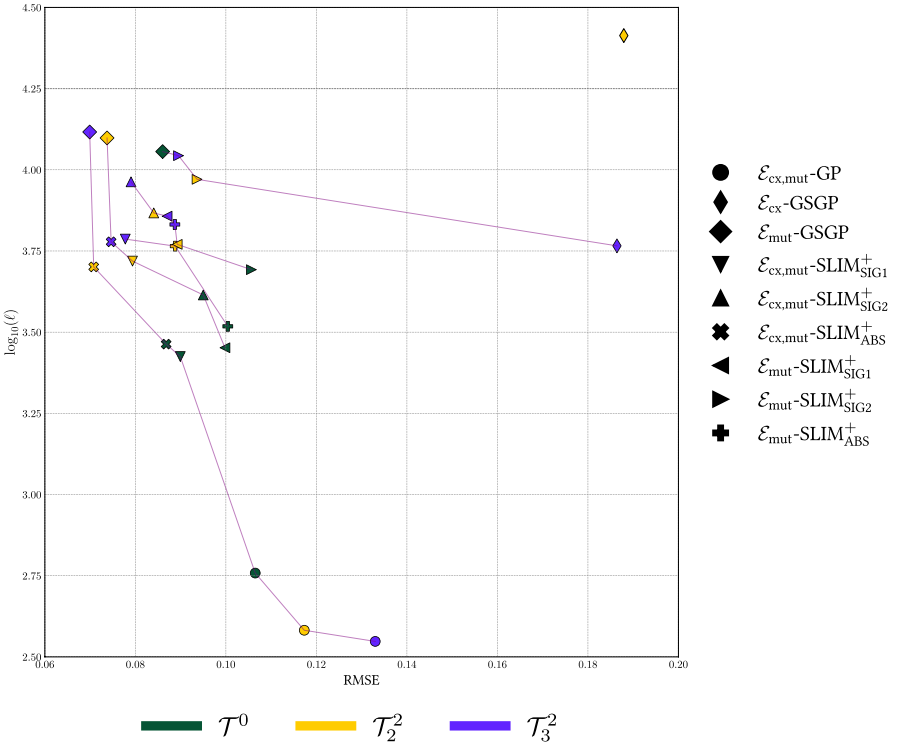

Fig. 14 Zoomed version of Fig. 9

Acknowledgements This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152/2020 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS (https://doi.org/10.54499/UIDB/04152/2020).
This research is partially supported by the PRIN 2022 PNRR project "Cellular Automata Synthesis for Cryptography Applications (CASCA)" (P2022MPFRT) financed by the European Union – Next Generation EU.

Author contributions L.B., D.F., G.P., and L.R. wrote the manuscript. L.B. and L.R. designed and implemented the cellular framework. D.F. and G.P. designed and implemented the SLIM framework. L.R. executed the experiments and created the visualizations. A.D.L., L.M., G.P., and L.V. supervised the experiments. L.M. and A.D.L. designed the cellular methodology. L.V. designed the SLIM methodology. All authors reviewed the manuscript.

Funding Open access funding provided by Università degli Studi di Trieste within the CRUI-CARE Agreement.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. J.V. Neumann, Theory of self-reproducing automata. Math. Comput **21**, 745 (1966)
2. E.F. Codd, *Cellular Automata* (Academic press, Cambridge, 1968)
3. P. Sarkar, A brief history of cellular automata. Acm Comput. Surv. (csur) **32**(1), 80–107 (2000)
4. E. Alba, B. Dorronsoro, The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans. Evol. Comput. **9**(2), 126–142 (2005)
5. E. Alba, B. Dorronsoro, *Introduction to Cellular Genetic Algorithms* (Springer, Boston, MA, 2008), pp.3–20
6. C. Salto, E. Alba, Cellular genetic algorithms: understanding the behavior of using neighborhoods. Appl. Artif. Intell. **33**(10), 863–880 (2019)
7. L. Bonin, L. Rovito, A. De Lorenzo, L. Manzoni, Cellular geometric semantic genetic programming. Genet. Program Evolvable Mach. **25**(1), 8 (2024)
8. T. Murata, M. Gen, Cellular genetic algorithm for multi-objective optimization. In: Proceedings of the 4th Asian Fuzzy System Symposium, pp. 538–542 (2002). Citeseer
9. A.J. Nebro, J.J. Durillo, F. Luna, B. Dorronsoro, E. Alba, Mocell: a cellular genetic algorithm for multiobjective optimization. Int. J. Intell. Syst. **24**(7), 726–746 (2009)
10. M. Zhang, N. Tian, V. Palade, Z. Ji, Y. Wang, Cellular artificial bee colony algorithm with gaussian distribution. Inf. Sci. **462**, 374–401 (2018)
11. M. Giacobini, M. Tomassini, A.G. Tettamanzi, E. Alba, Selection intensity in cellular evolutionary algorithms for regular lattices. IEEE Trans. Evol. Comput. **9**(5), 489–505 (2005)

12. G. Pietropolli, S. Nichele, E. Medvet, The role of the substrate in ca-based evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 768–777. ACM, Melbourne, Australia (2024)
13. P. Orzechowski, W.L. Cava, J.H. Moore, Where are we now? a large benchmark study of recent symbolic regression methods. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1183–1190. ACM, Kyoto, Japan (2018)
14. W. La Cava, P. Orzechowski, B. Burlacu, F.O. Franca, M. Virgolin, Y. Jin, M. Kommenda, J.H. Moore, Contemporary symbolic regression methods and their relative performance. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2021)
15. M. Virgolin, S.P. Pissis, Symbolic Regression is NP-hard (2022)
16. G. Folino, C. Pizzuti, G. Spezzano, A cellular genetic programming approach to classification. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1015–1020. Morgan Kaufmann, Orlando, FL, USA (1999)
17. A. Takac, Cellular genetic programming algorithm applied to classification task. Neural Netw. World **14**, 435–452 (2004)
18. A. Takac, Application of cellular genetic programming in data mining. In: Proceedings of the Conference on Knowledge Management (Knowledge). Citeseer, Ljubljana, Slovenia (2004)
19. G. Folino, C. Pizzuti, G. Spezzano, A scalable cellular implementation of parallel genetic programming. IEEE Trans. Evol. Comput. **7**(1), 37–53 (2003)
20. D.R. White, J. McDermott, M. Castelli, L. Manzoni, B.W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, S. Luke, Better gp benchmarks: community survey results and proposals. Genet. Program Evolvable Mach. **14**, 3–29 (2013)
21. A. Moraglio, K. Krawiec, C.G. Johnson, Geometric semantic genetic programming. In: Proceedings of the 12th International Conference on Parallel Problem Solving from Nature (PPSN). Lecture Notes in Computer Science, vol. 7491, pp. 21–31. Springer, Taormina, Italy (2012)
22. L. Vanneschi, An introduction to geometric semantic genetic programming. In: NEO 2015: Results of the Numerical and Evolutionary Optimization Workshop NEO 2015 Held at September 23-25 2015 in Tijuana, Mexico, pp. 3–42 (2016). Springer
23. L. Vanneschi, Slim_gsgp: The non-bloating geometric semantic genetic programming. In: Proceedings of the European Conference on Genetic Programming (EuroGP, Part of EvoStar). Lecture Notes in Computer Science, vol. 14474, pp. 125–141. Springer, Aberystwyth, UK (2024)
24. L. Vanneschi, D. Farinati, D. Rasteiro, L. Rosenfeld, G. Pietropolli, S. Silva, Exploring non-bloating geometric semantic genetic programming. Genetic Program. Theory Practice **XXI**, 237–258 (2025)
25. G. Pietropolli, D. Farinati, L. Manzoni, M. Castelli, S. Silva, L. Vanneschi, Introducing crossover in slim-gsgp. In: Proceedings of the European Conference on Genetic Programming (EuroGP, Part of EvoStar). Lecture Notes in Computer Science, vol. TBD, pp. 103–119. Springer, TBD (2025)
26. L. Rovito, L. Bonin, D. Farinati, L. Vanneschi, L. Manzoni, A.D. Lorenzo, G. Pietropolli, Exploring the integration of cellular structures in genetic programming-based methods. In: Proceedings of the European Conference on Genetic Programming (EuroGP, Part of EvoStar). Lecture Notes in Computer Science, vol. TBD, pp. 120–138. Springer, TBD (2025)
27. J.H. Holland, Genetic algorithms and the optimal allocation of trials. SIAM J. Comput. **2**(2), 88–105 (1973)
28. Y. Deng, J. Xiong, Q. Wang, A hybrid cellular genetic algorithm for the traveling salesman problem. Math. Probl. Eng. **2021**, 1–16 (2021)
29. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
30. L. Mariot, S. Picek, D. Jakobovic, A. Leporati, Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 306–313. ACM, Berlin, Germany (2017)
31. Y. Shi, H. Liu, L. Gao, G. Zhang, Cellular particle swarm optimization. Inf. Sci. **181**(20), 4460–4493 (2011)
32. M.A. Al-Betar, A.T. Khader, M.A. Awadallah, M.H. Alawan, B. Zaqaibeh, Cellular harmony search for optimization problems. J. Appl. Math. **2013**(1), 139464 (2013)
33. J.R. Koza, Genetic programming as a means for programming computers by natural selection. Stat. Comput. **4**(2), 87–112 (1994)
34. W. Banzhaf, J.R. Koza, C. Ryan, L. Spector, C. Jacob, Genetic programming. IEEE Intell. Syst. Appl **15**(3), 74–84 (2000)

35. M.T. Ahvanooey, Q. Li, M. Wu, S. Wang, A survey of genetic programming and its applications. KSII Trans. Internet. Info. (TIIS) **13**(4), 1765–1794 (2019)
36. V.K. Dabhi, S. Chaudhary, Empirical modeling using genetic programming: a survey of issues and approaches. Nat. Comput. **14**, 303–330 (2015)
37. Q. Huynh, H. Singh, T. Ray, A. Oyama, Improved genetic programming for symbolic regression: Case studies on practical applications. In: 2022 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1135–1142 (2022)
38. W. La Cava, K. Danai, L. Spector, Inference of compact nonlinear dynamic models by epigenetic local search. Eng. Appl. Artif. Intell. **55**, 292–306 (2016)
39. W. La Cava, K. Danai, L. Spector, P. Fleming, A. Wright, M. Lackner, Automatic identification of wind turbine models using evolutionary multiobjective optimization. Renew. Energy **87**, 892–902 (2016)
40. Y. Mei, Q. Chen, A. Lensen, B. Xue, M. Zhang, Explainable artificial intelligence by genetic programming: a survey. IEEE Trans. Evol. Comput. **27**(3), 621–641 (2023)
41. A. Lensen, B. Xue, M. Zhang, Genetic programming for evolving a front of interpretable models for data visualization. IEEE Trans. Cybern **51**(11), 5468–5482 (2021)
42. L.A. Ferreira, F.G. Guimarães, R. Silva, Applying genetic programming to improve interpretability in machine learning models. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2020)
43. T. Hu, *Genetic Programming for Interpretable and Explainable Machine Learning* (Springer, Singapore, 2023), pp.81–90
44. M. Virgolin, A.D. Lorenzo, F. Randone, E. Medvet, M. Wahde, Model learning with personalized interpretability estimation (ml-pie). In: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO Companion), pp. 1355–1364. ACM, Lille, France (2021)
45. G. Nadizar, L. Rovito, A. De Lorenzo, E. Medvet, M. Virgolin, 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), 1–30 (2024)
46. G. Nadizar, E. Medvet, D. Wilson, Searching for a diversity of interpretable graph control policies. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '24, pp. 933–941. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3638529.3653987
47. G. Nadizar, E. Medvet, D.G. Wilson, Naturally interpretable control policies via graph-based genetic programming, in *Genetic Programming*. ed. by M. Giacobini, B. Xue, L. Manzoni (Springer, Cham, 2024), pp.73–89
48. K.B. Rebuli, M. Giacobini, S. Silva, L. Vanneschi, A comparison of structural complexity metrics for explainable genetic programming. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO Companion), pp. 539–542. ACM, Lisbon, Portugal (2023)
49. M. Virgolin, T. Alderliesten, C. Witteveen, P.A. Bosman, Improving model-based genetic programming for symbolic regression of small expressions. Evol. Comput. **29**(2), 211–237 (2021)
50. W.N. Martin, Island (migration) models: evolutionary algorithms based on punctuated equilibria. Handbook of evolutionary computation (1997)
51. P.A. Whigham, G. Dick, Implicitly controlling bloat in genetic programming. IEEE Trans. Evol. Comput. **14**(2), 173–190 (2009)
52. G. Dick, P.A. Whigham, Controlling bloat through parsimonious elitist replacement and spatial structure. In: Proceedings of the 16th European Conference on Genetic Programming (EuroGP, Part of EvoStar). Lecture Notes in Computer Science, vol. 7831, pp. 13–24. Springer, Vienna, Austria (2013)
53. A.D. Cioppa, A. Marcelli, P. Napoli, Speciation in evolutionary algorithms: Adaptive species discovery. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 1053–1060. ACM, Dublin, Ireland (2011)
54. K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
55. L. Trujillo, L. Muñoz, E. Galván-López, S. Silva, Neat genetic programming: controlling bloat naturally. Inf. Sci. **333**, 21–43 (2016)
56. P. Juárez-Smith, L. Trujillo, M. García-Valdez, F. Vega, F. Chávez, Local search in speciation-based bloat control for genetic programming. Genet. Program Evolvable Mach. **20**, 351–384 (2019)
57. S. Cussat-Blanc, K. Harrington, J. Pollack, Gene regulatory network evolution through augmenting topologies. IEEE Trans. Evol. Comput. **19**(6), 823–837 (2015)

58. T.M. Martins, R.F. Neves, Applying genetic algorithms with speciation for optimization of grid template pattern detection in financial markets. Expert Syst. Appl. **147**, 113191 (2020)
59. R. Wickman, B. Poudel, T.M. Villarreal, X. Zhang, W. Li, Efficient quality-diversity optimization through diverse quality species. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO Companion), pp. 699–702. ACM, Lisbon, Portugal (2023)
60. D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of Genetic Algorithms vol. 1, pp. 69–93. Elsevier, San Mateo, CA (1991)
61. H. Xie, M. Zhang, Impacts of sampling strategies in tournament selection for genetic programming. Soft. Comput. **16**, 615–633 (2012)
62. J. Sarma, K. De Jong, An analysis of the effects of neighborhood size and shape on local selection algorithms. In: International Conference on Parallel Problem Solving From Nature, pp. 236–244 (1996). Springer
63. M. Castelli, L. Vanneschi, S. Silva, Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. Expert Syst. Appl. **40**(17), 6856–6862 (2013)
64. M. Castelli, L. Vanneschi, S. Silva, Prediction of the unified parkinson's disease rating scale assessment using a genetic programming system with geometric semantic genetic operators. Expert Syst. Appl. **41**(10), 4608–4616 (2014)
65. L. Vanneschi, S. Silva, M. Castelli, L. Manzoni, Geometric semantic genetic programming for real life applications. Genetic programming theory and practice xi, 191–209 (2014)
66. L. Vanneschi, M. Castelli, S. Silva, A survey of semantic methods in genetic programming. Genet. Program Evolvable Mach. **15**, 195–214 (2014)
67. L. Vanneschi, M. Castelli, L. Manzoni, S. Silva, A new implementation of geometric semantic gp and its application to problems in pharmacokinetics. In: Proceedings of the 16th European Conference on Genetic Programming (EuroGP, Part of EvoStar). Lecture Notes in Computer Science, vol. 7831, pp. 205–216. Springer, Vienna, Austria (2013)
68. T.P. Pawlak, B. Wieloch, K. Krawiec, Review and comparative analysis of geometric semantic crossovers. Genet. Program. Evolvable Mach. **16**, 351–386 (2015)
69. M. Castelli, L. Manzoni, I. Gonçalves, L. Vanneschi, L. Trujillo, S. Silva, An analysis of geometric semantic crossover: A computational geometry approach, pp. 201–208 (2016)
70. G. Nadizar, F. Garrow, B. Sakallioglu, L. Canonne, S. Silva, L. Vanneschi, An investigation of geometric semantic gp with linear scaling. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1165–1174. ACM, Lisbon, Portugal (2023)
71. G. Nadizar, B. Sakallioglu, F. Garrow, S. Silva, L. Vanneschi, Geometric semantic gp with linear scaling: darwinian versus lamarckian evolution. Genet. Program. Evolvable Mach. **25**(2), 1–24 (2024)
72. G. Pietropolli, L. Manzoni, A. Paoletti, M. Castelli, Combining geometric semantic gp with gradient-descent optimization, in *Genetic Program*. ed. by E. Medvet, G. Pappa, B. Xue (Springer, Cham, 2022), pp.19–33
73. G. Pietropolli, L. Manzoni, A. Paoletti, M. Castelli, On the hybridization of geometric semantic gp with gradient-based optimizers. Genet. Program. Evolvable Mach. **24**(2), 16 (2023)
74. M. Castelli, L. Manzoni, L. Vanneschi, S. Silva, A. Popovič, Self-tuning geometric semantic genetic programming. Genet. Program. Evolvable Mach. **17**, 55–74 (2016)
75. D. Farinati, I. Bakurov, L. Vanneschi, A study of dynamic populations in geometric semantic genetic programming. Inf. Sci. **648**, 119513 (2023)
76. L. Trujillo, J.M.M. Contreras, D.E. Hernandez, M. Castelli, J.J. Tapia, Gsgp-cuda-a cuda framework for geometric semantic genetic programming. SoftwareX **18**, 101085 (2022)
77. M. Castelli, S. Silva, L. Vanneschi, A c++ framework for geometric semantic genetic programming. Genet. Program. Evolvable Mach. **16**, 73–81 (2015)
78. M. Castelli, L. Vanneschi, A. Popovič, Controlling individuals growth in semantic genetic programming through elitist replacement. Comput. Intell. Neurosci. **2016**, 42–42 (2016)
79. J.F.B.S. Martins, L.O.V.B. Oliveira, L.F. Miranda, F. Casadei, G.L. Pappa, Solving the exponential growth of symbolic regression trees in geometric semantic genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1151–1158. ACM, Kyoto, Japan (2018)
80. D. Koga, K. Ohnishi, Non-generational geometric semantic genetic programming. In: Proceedings of the 2021 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7. IEEE, Orlando, FL, USA (2021)

81. L. Vanneschi, M. Castelli, , L. Castelli, S. Silva, A new implementation of geometric semantic gp and its application to problems in pharmacokinetics. In: Proceedings of the 16th European Conference on Genetic Programming (EuroGP, Part of EvoStar). Lecture Notes in Computer Science, vol. 7831, pp. 205–216. Springer, Vienna, Austria (2013)
82. W.B. Langdon, R. Poli, N.F. McPhee, J.R. Koza, Genetic programming: An introduction and tutorial, with a survey of techniques and applications. Computational intelligence: A compendium, 927–1028 (2008)
83. M. O'Neill, Riccardo poli, william b. langdon, nicholas f. mcphee: A field guide to genetic programming. Genet. Program. Evolvable Mach. **10**(2), 229–230 (2009)
84. L. Vanneschi, S. Silva, *Lectures on Intelligent Systems* (Springer, Cham, Switzerland, 2023)
85. I. Gonçalves, S. Silva, C.M. Fonseca, On the generalization ability of geometric semantic genetic programming, in *Genetic Program.* ed. by P. Machado, M.I. Heywood, J. McDermott, M. Castelli, P. García-Sánchez, P. Burelli, S. Risi, K. Sim (Springer, Cham, 2015), pp.41–52
86. I. Bakurov, J.M. Contreras, M. Castelli, N. Rodrigues, S. Silva, L. Trujillo, L. Vanneschi, Geometric semantic genetic programming with normalized and standardized random programs. Genet. Program. Evolvable Mach. **25**(1), (2024)
87. A. Moraglio, A. Mambrini, Runtime analysis of mutation-based geometric semantic genetic programming for basis functions regression. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 989–996. ACM, Amsterdam, The Netherlands (2013)
88. M. Castelli, L. Manzoni, I. Gonçalves, L. Vanneschi, L. Trujillo, S. Silva, An analysis of geometric semantic crossover: A computational geometry approach. In: IJCCI (ECTA), pp. 201–208 (2016)
89. R. Poli, W.B. Langdon, N.F. McPhee, J.R. Koza, Genetic programming: An introductory tutorial and a survey of techniques and applications. Univ. Essex School of Computer Science and Eletronic Engineering Technical Report No. CES-475, 1–112 (2007)
90. T.F. Brooks, D.S. Pope, M.A. Marcolini, Airfoil self-noise and prediction (1989)
91. I.-C. Yeh, Simulation of concrete slump using neural networks. Proc. Inst. Civil Eng. Construct. Mater. **162**(1), 11–18 (2009)
92. I. Ortigosa, R. Lopez, J. Garcia, A neural networks approach to residuary resistance of sailing yachts prediction. In: Proceedings of the International Conference on Marine Engineering (MARINE), vol. 2007. Barcelona, Spain, p. 250 (2007)
93. D. Ballabio, M. Cassotti, V. Consonni, R. Todeschini, QSAR aquatic toxicity. UCI Machine Learning Repository (2014)
94. J.S. Armstrong, F. Collopy, Error measures for generalizing about forecasting methods: empirical comparisons. Int. J. Forecast. **8**(1), 69–80 (1992)
95. P.A. Moran, Notes on continuous stochastic phenomena. Biometrika **37**(1/2), 17–23 (1950)
96. H. Li, C.A. Calder, N. Cressie, Beyond moran's i: testing for spatial dependence based on the spatial autoregressive model. Geogr. Anal. **39**(4), 357–375 (2007)
97. L. Anselin, S. Rey, Properties of tests for spatial dependence in linear regression models. Geogr. Anal. **23**(2), 112–131 (1991)
98. H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. **18**(1), 50–60 (1947)
99. S. Holm, A simple sequentially rejective multiple test procedure. Scandinavian journal of statistics, 65–70 (1979)
100. W.H. Kruskal, W.A. Wallis, Use of ranks in one-criterion variance analysis. J. Am. Stat. Assoc. **47**(260), 583–621 (1952)

Authors and Affiliations

Luigi Rovito[1] · Lorenzo Bonin[1] · Davide Farinati[3] · Leonardo Vanneschi[3] · Luca Manzoni[1] · Andrea De Lorenzo[2] · Gloria Pietropolli[1]

✉ Luigi Rovito
luigi.rovito@units.it

Lorenzo Bonin
lorenzo.bonin@phd.units.it

Davide Farinati
dfarinati@novaims.unl.pt

Leonardo Vanneschi
lvanneschi@novaims.unl.pt

Luca Manzoni
lmanzoni@units.it

Andrea De Lorenzo
andrea.delorenzo@units.it

Gloria Pietropolli
gloria.pietropolli@units.it

[1] Department of Mathematics, Informatics, and Geosciences, University of Trieste, Via Valerio 12, 34127 Trieste, IT, Italy

[2] Department of Engineering and Architecture, University of Trieste, Via Valerio 6, 34127 Trieste, IT, Italy

[3] Information Management School, Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, PT, Portugal