Soft Computing
https://doi.org/10.1007/s00500-025-11051-7

OPTIMIZATION

Local search, semantics, and genetic programming: a global analysis

Fabio Anselmi[1] · **Mauro Castelli**[2] · **Alberto d'Onofrio**[1] · **Luca Manzoni**[1] · **Luca Mariot**[3] · **Martina Saletta**[4]

Received: 30 January 2025 / Accepted: 13 December 2025

Abstract

Geometric Semantic Genetic Programming (GSGP) is a powerful variant of Genetic Programming (GP) that defines genetic operators inducing unimodal fitness landscapes. In recent years, a new mutation operator, Geometric Semantic Mutation with Local Search (GSM-LS), has been proposed to include a local search step in the mutation process. The core idea of GSM-LS is to incorporate a linear regression step during mutation, thereby accelerating convergence toward high-quality solutions. While GSM-LS helps the convergence of the evolutionary search, it is prone to overfitting. Thus, it was suggested to apply GSM-LS only for a limited number of generations and then revert to standard geometric semantic mutation. A more recently defined variant of GSGP (called GSGP-reg) also includes a local search step, but shares similar strengths and weaknesses with GSM-LS. Here, we investigate several strategies to mitigate overfitting in GSM-LS and GSGP-reg, ranging from simple regularized regression techniques to adaptive methods that estimate overfitting risk at each mutation. The latter approaches partition the training set into two subsets: one used to perform the mutation, and the other to evaluate the risk of overfitting based on the mutation's impact on held-out data. Experimental evaluations across seven real-world regression benchmarks show that, while plain GSGP underperforms on all datasets, methods incorporating local search often achieve significantly better test performance. For example, on the Airfoil dataset, the GSM-LS variant achieves a median RMSE below 10 compared to 30 with standard GSGP. On the LD50 and Bioavailability datasets, the proposed gen and ridge-regularized variants effectively mitigate overfitting, reducing test RMSE by up to 40% relative to baseline GSGP. We conclude that local search, when used with regularization strategies, enhances GSGP's performance and generalization capability across a diverse range of tasks.

Keywords Genetic programming · Semantics · Local search · Evolutionary computation

✉ Luca Mariot
l.mariot@utwente.nl

Fabio Anselmi
fabio.anselmi@units.it

Mauro Castelli
mcastelli@novaims.unl.pt

Alberto d'Onofrio
alberto.donofrio@units.it

Luca Manzoni
lmanzoni@units.it

Martina Saletta
martina.saletta@unibg.it

[1] Dipartimento di Matematica, Informatica e Geoscienze, Università degli Studi di Trieste, Via Alfonso Valerio 12/1, 34127 Trieste, TS, Italy

[2] NOVA Information Management School (NOVA IMS), Universidade NOVA de Lisboa, Campus de Campolide, 1070-312 Lisbon, Portugal

[3] Semantics, Cybersecurity and Services Group, University of Twente, Drienerlolaan 5, 7522 NB Enschede, The Netherlands

[4] Dipartimento di Scienze Umane e Sociali, Università degli Studi di Bergamo, Piazzale Sant'Agostino 2, 24129 Bergamo, Italy

Published online: 21 January 2026

1 Introduction

Genetic Programming (GP) (Koza 1994) was introduced 30 years ago as a method to evolve computer programs by mimicking the principles of Darwinian evolution. In its original version, GP uses genetic operators (crossover and mutation) to produce new individuals starting from a given population. These operators allow the creation of new solutions by operating on the syntax of the existing solutions. In other words, both crossover and mutation generate new children by acting on the structure of the parent solutions. In the most common implementation of tree-based GP, crossover produces two children by swapping parents' subtrees, while mutation replaces a parent's subtree with a randomly generated one.

While research demonstrated the suitability of standard syntax-based GP in addressing complex problems over different domains and reaching human-competitive results (Koza 2010), researchers started to analyze the properties of the evolutionary process. From this analysis, an important drawback related to syntax-based genetic operators emerged: knowing the outputs of the parents is not useful for predicting the outputs of the children. This is because the standard, syntax-based genetic operators work by only considering the structure of the parents and ignoring the results they produced over the available observations (Vanneschi et al. 2014). Even more important, the output of a GP solution is what matters to practitioners. Thus, it becomes evident that purely syntactic manipulations (those that do not meaningfully alter the semantics of individuals), are unlikely to improve the quality of already well-performing solutions. This limitation constrains the ability of GP to discover further improvements once a good solution has been found.

To overcome this limitation, GP research started investigating methods to include semantic awareness in the evolutionary process. Krawiec and Pawlak (2013) proposed the definition of semantics nowadays used in the GP literature by taking inspiration from the field of formal languages, where semantics describes the effects of program execution for all possible input data. GP evaluates programs over a finite training set of fitness cases, each being a pair consisting of an input and the corresponding desired program output. Thus, assuming that this set is the only available data to specify the output of the sought solution, a natural definition of semantics—which we consider in this paper—is the vector of outputs a program produces for a set of observations (Vanneschi et al. 2014). By adopting this definition, the semantics of an individual is a point in an n-dimensional space (i.e., the semantic space), where n is the number of fitness cases.

The first studies trying to include semantic awareness in the evolutionary process were based on a simple idea: the use of traditional syntax-based genetic operators followed by the evaluation of some semantic criteria on the resulting children to accept or reject the newly generated solutions (Vanneschi et al. 2014). Despite the satisfactory results achieved by these methods, they presented significant shortcomings. First, they were time-consuming due to the effort needed to evaluate the semantic criteria, and second, a high number of individuals were rejected. These limitations made the evolutionary process unbearably slow, thus limiting the use of this semantic criteria only to toy problems.

The situation changed in 2012 when Moraglio et al. (2012) proposed a method allowing the inclusion of semantic awareness in a direct manner. In particular, they defined genetic operators called geometric semantic operators (GSOs) that can directly act on the semantics of the parent individuals to produce new solutions. Additionally, GSOs induce a unimodal fitness landscape on any supervised problem (i.e., regression and classification) (Moraglio and Poli 2004). The resulting GP algorithm was called geometric semantic genetic programming (GSGP). GSGP is nowadays a hot topic in evolutionary computation, with many studies showing its ability to outperform syntax-based GP over different domains (Castelli et al. 2017a, b, 2015c; McDermott et al. 2014) and with many improvements and variants proposed over the years (Papa et al. 2017; Farinati et al. 2023; Bonin et al. 2024; Vanneschi et al. 2025). Although existing implementations (Castelli et al. 2015a; Castelli and Manzoni 2019; Trujillo et al. 2022) of GSGP allow for a faster evaluation of a candidate solution than syntax-based GP, the number of iterations needed to converge and the increasing size (i.e., number of nodes) of the GSGP trees make it difficult to use GSGP to address problems characterized by a vast amount of data, and when the interpretability of the model is a fundamental requirement.

A promising path to tackle this issue is the combination of GSGP with a local search strategy (Castelli et al. 2015d). The inclusion of a local search method within GSGP aims at speeding up the search process so that good-quality solutions are obtained in a limited number of iterations. Moreover, running GSGP for a limited number of iterations allows obtaining smaller solutions. Local search methods coupled with GP represent an important research area within evolutionary computation. While this approach is well-established in the broader field (e.g., the entire research area of *memetic algorithms* (Neri and Cotta 2012)), it still remains comparatively underexplored in the context of GP.

This paper continues this investigation focusing on GSGP. In particular, this work is the natural consequence of two studies in which a local search optimizer was coupled with GSGP: in the first study (Castelli et al. 2015d), the

authors modified the geometric semantic mutation operator to incorporate a local search operator, resulting in a new mutation called GSM-LS; in the second study (Castelli et al. 2019) the authors presented promising results achieved by applying a new local search operator (GSGP-reg) to all the individuals during a separate step after mutation and crossover. Despite the satisfactory results achieved in both studies, GSGP with local search may result in severe overfitting in some of the considered problems. Thus, in this research, we investigate a strategy to limit the overfitting of GSM-LS and GSGP-reg, ranging from adaptive methods to estimate the risk of overfitting at each mutation to a simple regularized regression. We compare two techniques to include local search in GSGP and present results that may help practitioners to use GSGP with local search limit overfitting through regularization.

The paper is organized as follows: Section 2 recalls the definition of GSOs and previous work on local search methods within GSGP; Section 3 describes the strategy proposed in this paper to counteract overfitting; Section 4 outlines the benchmark problems considered in this study and the experimental settings; Section 5 discusses the results by comparing the performance of different local search methods. Finally, Section 6 concludes the paper by summarizing the main contributions of this work.

2 Background

In this section, we first recall the definitions of the geometric semantic operators used in this work. Subsequently, we outline existing approaches to integrate local search in standard GP and GSGP , and we conclude by recalling recent contributions related to our work.

2.1 Geometric semantic operators

In 2012, Moraglio et al. (2012) introduced Geometric Semantic Operators (GSOs), genetic operators that directly incorporate semantic awareness in the evolutionary process. Such GSOs produce a transformation on the syntax of the individuals that correspond to geometric crossover and ball mutation (Krawiec and Lichocki 2009) in the semantic space. More in detail, the geometric crossover generates only one child that stands, in the semantic space, between the two parent solutions. On the other hand, the geometric semantic mutation produces a new solution whose coordinates in the semantic space correspond to a slight perturbation of some of the coordinates of the parent solution.

In this study, we will focus on regression problems. Thus, we report the definition of the GSOs for real function domains. For further details of GSOs in different domains, the reader is referred to Moraglio et al. (2012).

Geometric Semantic Crossover (GSC). Given two parents represented as functions $T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic crossover returns the real function $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$, where T_R is a random real function whose output values range in the interval [0, 1]. As a results, for each $x \in \mathbb{R}^n$, if $T_1(x) \leq T_2(x)$ then $T_1(x) \leq T_{XO}(x) \leq T_2(x)$.

Geometric Semantic Mutation (GSM). Given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic mutation with mutation step *ms* returns the real function $T_M = T + ms \cdot (T_{R1} - T_{R2})$, where T_{R1} and T_{R2} are random real functions.

As shown in Castelli et al. (2016a); Enríquez-Zárate et al. (2017), limiting the codomain of T_{R1} and T_{R2} to [0, 1] helps improving the generalization ability of GSGP. With this constraint, for all $x \in \mathbb{R}^n$, we have that $|T(x) - T_M(x)| \leq ms$, i.e., T_M is a point in the semantic space that lies on a ball of radius *ms* centered in *T*.

2.2 Local search and GP

Coupling evolutionary algorithms (EAs) with local search strategies has been investigated in different studies (Chen et al. 2011; Neri and Cotta 2012; Črepinšek et al. 2013; Gao et al. 2019, 2021). The fundamental idea shared by these methods (often called memetic algorithms) is to fully explore the local region around each individual. By exploiting this principle, memetic algorithms can achieve better performance over the standard evolutionary approach (Neri and Cotta 2012; Chen et al. 2011). Despite these promising results, the literature features a relatively low number of contributions that couple GP with local search (Trujillo et al. 2018). In particular, the two main approaches to integrate a local search (LS) strategy into GP are: (1) apply LS on the syntax (Azad and Ryan 2014; Eskridge and Hougen 2004); or (2) apply it on numerical parameters of the program (Topchy and Punch 2001; Zhang and Smart 2004). In both cases, existing contributions showed that the strategy used to apply local search in GP has a significant impact on the performance of the resulting model. Focusing on the use of GP to address symbolic regression problems, Z-Flores and coauthors (Emigdio et al. 2014) suggested that the best strategy consists of the application of LS to all the individuals in the population or, eventually, to a subset of the best individuals. Moreover, the same study empirically showed that including an LS strategy improves convergence and performance while reducing the growth of the solutions' size.

Focusing on GSGP, the integration of a local search strategy in the search process is a recent topic. To the best of

our knowledge, the first contribution in this research strand was proposed by Castelli et al. (2015d) in 2015, where a greedy LS method was integrated into the geometric semantic mutation operator. In particular, given a parent individual T, the mutation operator with local search (GSM-LS) is defined as follows:

$$T_M = \alpha_0 + \alpha_1 \cdot T + \alpha_2 \cdot (T_{R1} - T_{R2})$$

where T_{R1} and T_{R2} are random trees with output in [0, 1], $\alpha_i \in \mathbb{R}$. Once the two random trees T_{R1} and T_{R2} have been selected, the mutation step consists in finding the values of α_0, α_1, and α_2 that minimize the sum of squared differences between the predicted outputs of the mutated tree and the target outputs. This corresponds to a linear regression problem, which can be solved exactly using the linear least squares method (Nievergelt 1994). Furthermore, if the output of the parent tree *T* has already been computed (e.g., during a previous fitness evaluation), the computation of α_0, α_1, and α_2 can be performed without re-evaluating *T*. However, it is worth noting that although this process may not require additional fitness evaluations, solving the least squares problem still involves computing the solution to a linear system, which may be computationally non-trivial depending on the size of the training data. Once the optimal coefficients are determined, the mutation is applied by replacing the parent tree *T* with the new, linearly combined tree.

As reported by Castelli et al. (2015d), the GSM-LS operator aims at determining the best linear combination of the parent tree and the random trees (T_{R1} and T_{R2}), and it is local in the sense of the linear problem it defines. The GSM-LS operator was considered in different studies (Castelli et al. 2015b; Hajek et al. 2019; Trujillo et al. 2018) and demonstrated its suitability in speeding up the convergence of the search process, thus reducing the size of the resulting solution. However, severe overfitting appeared in some of the investigated benchmarks, suggesting that further research is needed to fully understand how GSM-LS can improve the performance of GSGP without hampering its generalization ability. Already in the original work (Castelli et al. 2015d), the authors addressed this issue by applying GSM-LS only during the first 10 generations. This early stopping strategy was empirically shown to preserve most of the benefits of local search (such as faster convergence and improved training performance) while limiting the risk of overfitting. The initial proposal of GSM-LS presented in Castelli et al. (2015d) was subsequently extended in Castelli et al. (2019), where the authors introduced GSGP-reg (or reg for short), a GSGP algorithm that uses an alternative local search operator. This approach differs from GSM-LS by defining a generic set of functions that locally modify a candidate GP individual. The main idea is that the possibility of modifying the set of functions allows the creation of a problem-specific local search operator that should improve GSGP 's performance. In particular, let $f_1, \ldots, f_k$ be a set of (possibly non-linear) functions. Let us define the mutated tree $T'_\alpha(x) = \alpha_1 f_1(T(x)) + \cdots + \alpha_k f_k(T(x))$ where T is the parent tree and the α_1,..., α_k are scalar coefficients. Finding the optimal values of α_i that minimize the error on the training set is a linear regression problem, which can be solved with the same methods used for GSM-LS. In Castelli et al. (2019), the family of functions consisted of the identity (i.e., $f_1(x) = x$), the constant 1 (i.e., $f_2(x) = 1$), the positive part (i.e., $f_3(x) = \max(0, x)$), and negative part (i.e., $f_4(x) = \min(0, x)$). GSGP-reg achieved a performance comparable to GSGP with GSM-LS. However, as the authors pointed out, the problems in which the two GSGP variants tend to overfit are different, thus suggesting the need for further investigation in this area.

2.3 Recent advances in geometric semantic genetic programming

As mentioned in the Introduction, GSGP has been the subject of significant research interest due to its ability of inducing unimodal fitness landscapes and guaranteeing that the semantics of the offspring lie within bounded regions of their parents. A recent special issue of *Genetic Programming & Evolvable Machines* (Moraglio et al. 2025) marked the 10-year anniversary of GSGP and presented a comprehensive set of novel contributions aimed at overcoming known limitations. In what follows, we summarize the key contributions of this special issue concerning GSGP.

Bakurov et al. (2024), taking inspiration from the use of batch normalization in deep learning, proposed normalizing and standardizing the random functions used in semantic mutation, reducing the size of the resulting programs. Zhang et al. (2024) introduced a macro-crossover operator for multi-tree representations that enables expressive recombination in multivariate symbolic regression. In particular, multi-tree genetic programming with the geometric semantic macro-crossover operator significantly outperformed 22 machine learning algorithms over a state-of-the-art regression benchmark. Dick (2024) reinterpreted GSGP through an ensemble learning lens, offering theoretical insights into the robustness observed empirically and showing that the method can outperform gradient boosting. Bonin et al. (2024) developed a cellular GSGP variant where spatial constraints help preserve diversity and delay overfitting.

These developments expand the GSGP framework and inform our current approach, which focuses on combining semantic variation with controlled local refinement to enhance generalization.

Beyond the special issue summarized above, Pietropolli et al. (2022) recently proposed the use of an LS optimizer (i.e., Adam Kingma and Ba 2014) that acts on both GSOs introduced in Moraglio et al. (2012). Their idea is the following: after performing the evolutionary steps involving GSM and GSC, a new population $T = (T_1, T_2, \cdots, T_M)$ of M individuals is obtained, with T composed of differentiable functions (as they are obtained through additions, multiplications, and compositions of differentiable functions). By considering GSC defined as $T_{XO} = (T_1 \cdot \alpha) + ((1 - \alpha) \cdot T_2)$ (with $0 \leq \alpha \leq 1$) and GSM as $T_M = T + ms \cdot (T_{R1} - T_{R2})$ (with $0 \leq ms \leq 1$ and T_{R1}, T_{R2} random trees), and given that the values of α and *ms* are randomly initialized, it is possible to derive T with respect to α, $\beta = (1 - \alpha)$, and *ms*. Therefore, the gradient-based optimizer can be applied considering $\theta = (\alpha, \beta, ms)$ as the parameters vector. While the resulting algorithm can be considered as a local search, it deviates from GSM-LS and GSGP-reg by performing non-local modifications to the tree representation. In fact, not only do the parameters to be optimized modify the (parameters of) crossovers and mutations, but due to the representation of GSGP trees as a directed acyclic graph, the optimization is performed on the entire population, at the same time. Due to this significant difference, we will not consider this approach in this work.

Another relevant contribution was proposed by Gonçalves et al. (2015) starting from the observation that the fixed (or random) step sizes used in the GSM operator can slow down the evolutionary search. To address this, Gonĉalves et al. proposed an "optimal mutation step" method that analytically computes the mutation magnitude to minimize training error. This introduces a deterministic local refinement within the stochastic mutation process, improving convergence.

The GSM-LS operator by Castelli et al. (2015d) implements this idea via linear regression, allowing the generation of semantically informed offspring. However, while effective, GSM-LS is susceptible to overfitting, an issue our study mitigates using validation-based filtering and ridge regression.

Beyond these algorithmic developments, recent work has also established important theoretical foundations for GSGP. Corus and Oliveto (2024) provided the first rigorous theoretical analysis of the generalization performance of GSGP for Boolean functions, demonstrating that geometric semantic operators can learn target concepts efficiently while maintaining good generalization properties. Their results offer theoretical justification for the empirical success of GSGP and provide insights into when and why geometric semantic operators outperform traditional GP.

More broadly, the field of evolutionary computation has seen significant advances in theoretical understanding. Recent surveys (Doerr and Neumann 2021) have synthesized progress in the theory of evolutionary algorithms for discrete optimization, while, in the book edited by Doerr and Neumann (2019), an overview of the state of the art concerning the complexity analysis of GP is presented (Lissovoi and Oliveto 2019). Finally, Doerr and Doerr (2019) reviewed existing contributions focused on the theory of parameter control for discrete black-box optimization.

2.4 Use of validation sets in genetic programming

Although rarely integrated into GP evolution loops, validation sets play a crucial role in model generalization. Gagné et al. (2006) explored their use in guiding evolution to avoid overfitting. The authors compared validation- and training-based selection and showed that validation-informed evolution promotes parsimony and robustness. Žegklitz and Pošík (2015) presented a study focused on overfitting and model selection. They evaluated several ways of dealing with overfitting, based on a random sampling technique and on using a validation set, all with an emphasis on model selection. Experimental results, achieved over artificial and real–world datasets, showed good performance of the baseline approach which uses the full training data, thus contradicting previous works (Gonçalves and Silva 2013). More recently, Rivero et al. (2020) highlighted a distinctive feature of GP: in contrast to many traditional machine learning techniques, GP evolves a population of models rather than training a single solution. As a consequence, the use of the validation dataset has several possibilities because any of those evolved models could be evaluated. Moving from this observation, Rivero et al. explore the usage of the validation dataset not only on the training-best individual but also in a subset with the training-best individuals of the population. Experimental results demonstrated the beneficial effects of using the validation dataset on a subset of the population instead of only on the training-best individual.

Inspired by these works, our proposed GSGP-gen mechanism uses a held-out subset to determine whether a local refinement improves unseen performance, guiding both model acceptance and adaptive application frequency.

2.5 Adaptive parameter control

Parameter control in evolutionary algorithms has long been studied (Eiben et al. 2002). Unlike parameter tuning, which involves fixing parameter values prior to execution, parameter control enables dynamic adjustment of parameters during the evolutionary process. Parameter control can be grouped into two categories. Deterministic control adjusts parameters according to fixed, predefined rules without considering the current state of the search. Adaptive control modifies parameters, including the choice of local or global strategies, using feedback from the ongoing evolutionary process, such as trends in fitness improvement or population diversity.

Recent theoretical work (Doerr and Neumann 2019) has provided formal frameworks for understanding why parameter control mechanisms improve evolutionary algorithm performance. These results suggest that adaptive control is particularly beneficial when the optimal parameter settings vary across different stages of the search process or across different regions of the fitness landscape, conditions that commonly arise in GSGP when balancing exploration and exploitation through local search. These theoretical contributions inform our adaptive local search mechanism which dynamically adjusts the probability of applying local search based on performance feedback, a form of parameter control with theoretical grounding in adaptive optimization strategies.

The concept of combining global and local search strategies has also been formalized within the broader class of memetic algorithms, introduced by Moscato et al. (1989). Memetic algorithms incorporate a local improvement step within an evolutionary framework, often triggered probabilistically or based on certain fitness thresholds. In most implementations, the probability of invoking local search is fixed. However, an adaptive trigger (such as validation set feedback) allows for a more fine-grained control, preventing unnecessary local exploitation when generalization does not improve (Neri and Cotta 2012).

In GP, adaptive parameter control showed promise in improving performance and avoiding issues like premature convergence. For instance, Gregor and Spalek (2016) introduced mechanisms like Adaptive Value Setting of Mutation Rate (AVSMR) and flood strategies to dynamically adjust mutation rates based on fitness stagnation, helping the search process escape local optima. Similarly, Rost et al. (2016) proposed a reinforcement learning-based controller with adaptive discretization for parameter selection, demonstrating improved efficiency in evolutionary algorithms.

A systematic review by Aleti and Moser (2016) analyzed various adaptive parameter control methods, highlighting the importance of feedback mechanisms and the challenges in designing effective control strategies. Their work emphasizes the need for adaptive methods that can respond to the dynamic nature of the search process GP and other evolutionary algorithms.

Recently, Russo (2020) presented a method that allows for self-adapting almost all the internal parameters in parallel distributed client-server genetic programming. The approach is inherently evolutionary, effectively creating a two-level evolutionary framework. At the upper level, a traditional evolutionary algorithm operates with its own selection, crossover, and mutation mechanisms. Meanwhile, the lower level consists of a parallel-distributed software system for the formal modelling of numerical data, based on a hybrid neural-genetic programming technique.

To date, little work has been done to bring adaptive strategies into the domain of semantically aware GP. Castelli et al. (2016b) proposed an adaptive method that works by assigning a (different) crossover and mutation probability to each individual of the population. Experimental results demonstrated the effectiveness of this method, achieving a performance that was significantly superior to that of the baseline GSGP.

Building on this foundation, our work introduces a novel adaptive control mechanism guided by validation-set performance. Rather than applying local search uniformly, we adapt the probability of applying local refinement based on its success rate in improving generalization. This is achieved by splitting the training data into disjoint subsets and assessing the benefit of local search on unseen validation data. If local search leads to consistent validation improvements, its application probability increases; otherwise, it is downregulated.

To summarize, this study expands the literature on local search in GSGP by comparing two strategies and integrating novel mechanisms to prevent overfitting. By adaptively determining the application of local search using validation-based success rates and regularizing the semantic refinements via ridge regression, we create a hybrid model that balances exploitation and generalization. Unlike previous work, this is the first comprehensive effort to unify semantics, validation-aware local search, and adaptive control

into a GSGP framework and to benchmark its performance across multiple real-world regression problems.

3 Adaptive local search

Since one of the main problems of local search in GSGP is the risk of overfitting, a reasonable approach would be to test whether a local search step produces a solution that generalizes well before accepting it. This can be done by splitting the training set into two partitions, performing a local search step by using the individuals belonging to a partition, and then testing if the fitness of the individuals in the other partition improves. If the assessment shows poor generalization ability, the local search step is rejected, and the original individual is returned. The information concerning the fraction of accepted solutions obtained with the LS application can be used to change the probability of applying it. That is, if most local search steps result in overfitting individuals, it would be better to reduce the probability of applying a local search step. Those two principles are encapsulated in the "gen " adaptive local search algorithm presented below.

Formally, we can describe the outlined approach as follows. Let us consider a GSGP tree T evaluated on a training set X consisting of $m \in \mathbb{N}$ fitness cases. Let ℓ be a local search operator returning a tree $T' = \ell(T; X)$; notice that the local search step might depend on the training set X that can be employed in building T', as in GSM-LS and GSGP-reg. From now on, the fitness of T on the set X will be denoted as $f(T; X)$.

This general idea of the adaptive local search mechanism gen (the name stems from generalization-based) is encapsulated in two algorithms. The first one accepts the local search only if it is expected to give better fitness on unseen data, as described in Algorithm 1.

1: $X = X_1 \cup X_2$ s.t. $X_1 \cap X_2 = \emptyset$
2: Apply the local search operator ℓ on X_1 to obtain $T' = \ell(T; X_1)$
3: **if** $f(T'; X_2)$ is improved wrt $f(T; X_2)$ **then**
4: **return** T'
5: **else**
6: **return** T
7: **end if**

Algorithm 1 Local search is accepted only if it generalizes well on unseen data

We expect that if ℓ is applied and the tree T' is accepted, then T' will perform well on unseen data, thus avoiding overfitting. If, instead, T is returned, we expect T' to overfit. In our current implementation, we set $|X_1| = \lceil 0.9|X| \rceil$ and $|X_2| = \lfloor 0.1|X| \rfloor$.

Notice that some additional information can be obtained from multiple applications of the previous algorithm. If the individuals produced by the local search are rejected with a high frequency, the current population may be in a part of the search space where the local search operation ℓ is not beneficial. Hence, the second component of the gen algorithm is to apply the local search to each tree with a certain probability.

In particular, let t be the current generation and let $N_{\text{acc}}(t-1)$ be the number of accepted local search operations (i.e., local search operations resulting in the acceptance of T') up to generation $t-1$ over a total of $N(t-1)$ tries. Then, at the current generation, the probability of performing a local search for a tree is $\max\left\{0.01, \frac{N_{\text{acc}}(t-1)}{N(t-1)}\right\}$. Notice that there is a minimum probability of 0.01 for the local search to be applied, thus avoiding the removal of the local search altogether.

Now, let $n_{\text{acc}}(t)$ be the number of accepted local search operations for generation t and $n(t)$ the number of local search operations executed at generation t. Then, the cumulative number of accepted and executed local search operations up to generation t are respectively $N_{\text{acc}}(t) = N_{\text{acc}}(t-1) + n_{\text{acc}}(t)$ and $N(t) = N(t-1) + n(t)$. This means that the resulting probability will progressively change less, because the influence that a generation can have on its variation decreases over time.

4 Experiments

In this section, we define the objective of our experiments, the underlying settings and datasets, and the methods considered in our investigation.

First of all, there are three main research questions that we want to address in the experimental phase:

RQ1 Are the overfitting control strategies for local search able to prevent overfitting?

RQ2 Are these strategies still able to provide better performances than plain GSGP?

RQ3 Is there one strategy that can provide better performances than the others?

To answer these research questions, we will compare the performances of the different algorithms on seven different regression problems. In particular, for RQ3, we will consider several combinations of different approaches:

- the use of local search only in the first few generations;
- the adaptive search algorithm gen;
- the use of linear regression with L_2 regularization in the local search step.

4.1 Tested algorithms

Three baseline algorithms are considered and, for each of them, one or more overfitting-preventing techniques are applied, namely:

- Use the local search method—either GSM-LS or reg —only for the first ten generations. This is the original approach proposed for both techniques;
- Application of the gen adaptive local search method;
- Use ridge regression (Marquardt and Snee 1975) instead of the least squares linear regression with no regularization. In ridge regression, an additional term provides an L_2 regularization to avoid overfitting.

The addition of ridge regression as an overfitting prevention method is because overfitting—and modeling noise or outliers in the data—is a risk even for a simple linear regression. Thus, the use of a regularization factor limits the risk of overfitting which is a problem with using the standard linear least squares regression which has no regularization term.

We consider GSGP, GPLS (i.e., GSGP with the GSM-LS operator), and reg as the baseline algorithms. A combination of these algorithms with the different overfitting prevention strategies leads to the assessment of the following variants:

- GSGP. This is the standard GSGP algorithm where no local search step is performed. This is the algorithm as defined in Moraglio et al. (2012) with the implementation described in Vanneschi et al. (2013).
- GPLS. In this case, for all the generations, the mutation operator is GSM-LS.
- GPLS_r. The regression used inside GSM-LS is ridge regression.
- GPLS^g. The mutation operator GSM-LS uses the adaptive search gen.
- GPLS^g_r. In this case, the mutation operator GSM-LS uses the adaptive search gen, and the regression employed is ridge regression.
- hybrid. Like GPLS, but the GSM-LS operator is used only in the first 10 generations.
- hybrid_r. Like GPLS_r, but the GSM-LS operator is applied only for the first 10 generations.
- reg − full. In this case, the local search defined in reg is applied in all generations.
- $\text{reg} - \text{full}_r$. Like reg − full, but the regression used is ridge regression.
- reg. The originally defined reg where the search is applied only in the first 10 generations.
- reg_r. Like reg, but ridge regression is employed.
- reg^g. Like reg but using gen as an overfitting prevention technique.
- reg^g_r. The combination of reg_r (use of ridge regression) and reg^g (use of gen) at the same time.

As one can see, a subscript *r* indicates the use of *ridge regression*, while a superscript *g* represents the use of the gen algorithm.

4.2 Experimental settings

Algorithm 2 provides the complete pseudocode for the main GSGP evolutionary cycle (with auxiliary function outlined in Algorithm 3), where the mutation can either be GSM or GSM-LS. The algorithm clarifies several implementation details: the population size N is set to 100 individuals throughout all experiments; the evolutionary process is generational, meaning that N offspring are created in each generation; and elitism is implemented by combining the best individual from the previous generation with the N newly created offspring, then selecting the N best individuals for the next generation. This ensures that the best solution found so far is never lost during evolution.

Table 1 Number of input variables and instances for each dataset used in the experimental phase

	airfoil	bioav	concrete	parkinson	ppb	slump	LD50
# variables	5	241	8	18	628	9	6
# instances	1502	359	1029	5875	131	102	307

Require: Training set X, population size N, number of generations G, tournament size τ, mutation probability p_m, crossover probability p_c
Ensure: Best individual found

// Initialization
$P_0 \leftarrow$ Initialize population of N individuals using ramped-half-and-half (max depth 6)
Evaluate fitness of all individuals in P_0 on training set X
$best \leftarrow$ individual with best fitness in P_0
// Evolution
for $t = 1$ to G **do**
  $offspring \leftarrow \emptyset$
  // Create N offspring
  for $i = 1$ to N **do**
    // Tournament selection
    $parent_1 \leftarrow$ TournamentSelection(P_{t-1}, τ)
    $r \leftarrow$ random value in $[0, 1]$
    if $r < p_c$ **then**
      // Apply crossover
      $parent_2 \leftarrow$ TournamentSelection(P_{t-1}, τ)
      $child \leftarrow$ ApplyGeometricSemanticCrossover$(parent_1, parent_2)$
    else if $r < p_m + p_c$ **then**
      // Apply mutation (GSM or GSM-LS)
      if using GSM-LS **then**
        $child \leftarrow$ ApplyLocalSearchMutation$(parent_1, X)$
      else
        $child \leftarrow$ ApplyGeometricSemanticMutation$(parent_1)$
      end if
    else
      // Reproduction (copy parent)
      $child \leftarrow parent_1$
    end if
    $offspring \leftarrow offspring \cup \{child\}$
  end for
  // Evaluate offspring
  Evaluate fitness of all individuals in $offspring$ on X

  // Elitism: combine best from previous generation with offspring
  $combined \leftarrow \{best\} \cup offspring$

  // Select N best individuals for next generation
  $P_t \leftarrow$ SelectBest$(combined, N)$

  // Update best individual
  $best \leftarrow$ individual with best fitness in P_t
end for

return $best$

Algorithm 2 Main GSGP Algorithm with GSM or GSM-LS

Function ApplyLocalSearchMutation(T, X):
  // For GSM-LS:
  $T_M = \alpha_0 + \alpha_1 \cdot T + \alpha_2 \cdot (T_{R1} - T_{R2})$
  // For reg:
  $T'_\alpha(x) = \alpha_1 \cdot f_1(T(x)) + \cdots + \alpha_k \cdot f_k(T(x))$
  Generate random trees T_{R1}, T_{R2} (for GSM-LS) or define functions $f_1, \ldots, f_k$ (for reg)
  Solve linear regression to find optimal coefficients α minimizing error on X
return mutated tree with optimal coefficients

Algorithm 3 Local Search Mutation

For all tested methods, the set of functional symbols is $\{+, -, \times, \div\}$, where $\div$ is protected by returning 1 when the denominator is sufficiently close to 0. The set of terminal symbols is the set of input variables of the problems used in the benchmarks, with no fixed constants belonging to the terminal set. The trees in the initial generation (line 2 of Algorithm 2) and the random trees used by the semantic operators were all generated with a ramped-half-and-half initialization method with a maximum depth of 6. During the evolution, the selection method used was tournament selection (lines 11 and 15 of Algorithm 2) with a tournament size $\tau = 4$, while mutation and crossover were performed, in an exclusive way, with probabilities of $p_m = 0.6$ and $p_c = 0.4$, respectively (lines 15 and 24 of Algorithm 2).

In the case of the gen method applied to GSM-LS, the two probabilities were combined multiplicatively (i.e., once an individual has been selected for mutation, it can either be subject to a local search step or remain unchanged). For all methods elitism is used, with the best performer on the training set at one generation surviving to the next one. For the methods based on reg, the function used in the regression were the tree $f_1(x) = T(x)$, $f_2(x) = 1$, $f_3(x) = \min\{0, T(x)\}$ (i.e., the negative part), and $f_4(x) = \max\{0, T(x)\}$ (i.e., the positive part), with $T(x)$ representing the tree, on which the local search is to be applied, evaluated on input x. When ridge regression is used, the regularization factor is set to 0.001; tests with values of 0.01 and 0.0001 were also performed with no change in the trends of the results. Thus, to simplify the discussion, only the results for 0.001 are shown.

The population size is set to 100 individuals, evolved for 100 generations. Notice that, while the solution of a linear regression problem, whether via ordinary least squares or ridge regression, incurs a non-negligible computational cost, it does not require to reevaluating the tree. As such, this step is not counted in terms of fitness evaluations. Preliminary measurements showed that the slowdown due to the solution of the linear regression task was limited. For each dataset, 100 independent runs are performed with a different random split between training (70% of the samples) and test sets (30% of the samples). In each run, the fitness on the training and test set of the best individual, computed as the root mean squared error (RMSE), was recorded for each generation. For all the methods using the gen algorithm, the probability of performing a local search step is also recorded at each generation. To compare the results at the last generation, the Mann-Whitney U-test (with Bonferroni correction) is used since it makes no assumption on the distribution of the underlying samples.

4.3 Datasets

To perform a comparison of the different methods, we used 7 different symbolic regression problems, with some of them widely used in the community (McDermott et al. 2012; White et al. 2013). They are all real-world data collected from multiple domains, as detailed below:

- *Airfoil*. The dataset consists of different airfoils at various wind tunnel speeds and angles of attack, with the target being the scaled sound pressure level in decibels.
- *Human Oral Bioavailability* (bioav). This dataset has a pharmacokinetic parameter as a target, which measures the quantity of orally administered drug that enters the blood circulation after being processed by the liver. Each instance consists of 241 molecular descriptors.
- *Concrete Compressive Strength* (concrete). The target is a parameter measuring the resistance to compression of a particular mix of concrete. Each instance is a different concrete mix described by 6 variables.
- *Parkinson*. The dataset consists of voice measurements of 42 patients in the early stage of Parkinson's disease recruited for a six-month trial of symptom progression monitoring. The input variables consist of patient information and 16 voice measurements. The target is to predict the total UPDRS (Unified Parkinson's Disease Rating Scale) score.
- *Plasma Protein Binding* (ppb). Plasma Protein Binding is a parameter measuring the quantity of drug that reaches circulation and further attaches to plasma proteins in the blood. The dataset is composed of 131 molecule instances, each described by 626 features.
- *Concrete Slump* (slump). The target variable, i.e., concrete slump, is a parameter that measures the consistency of fresh concrete. Each instance is a concrete mix described by 8 variables.
- *Median Lethal Dose* (LD50). The Median Lethal Dose measures the quantity of a drug necessary to kill half of the test organisms. This dataset is obtained by considering mice as test organisms and oral supplying as an administration route. Each instance represents a molecule described by 626 features.

The number of instances and the number of variables for each dataset are summarized in Table 1. For more details on the datasets, we refer the reader to Castelli et al. (2013); Yeh (1998) for the concrete and slump problems, to Archetti et al. (2007) for all the pharmacokinetic datasets, to Tsanas et al. (2009) for the Parkinson dataset and to Brooks et al. (1989) for the airfoil problem.

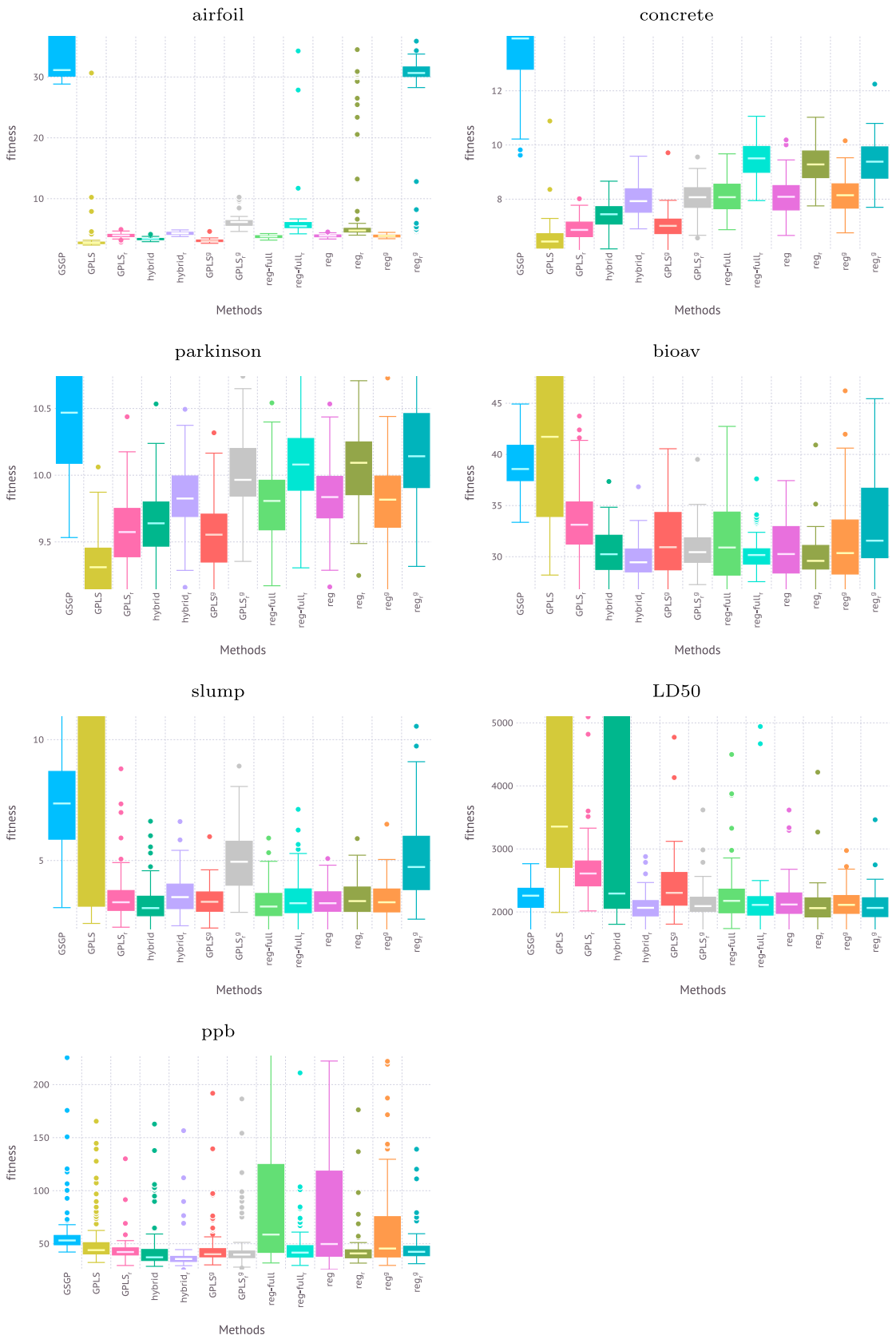

Fig. 1 Boxplots of the fitness at the last generation on the *test set* for all the considered problems. The boxes show the second and third quartiles, with the internal line showing the median. The whiskers show the first and fourth quartiles, excluding the outliers that are explicitly shown as points

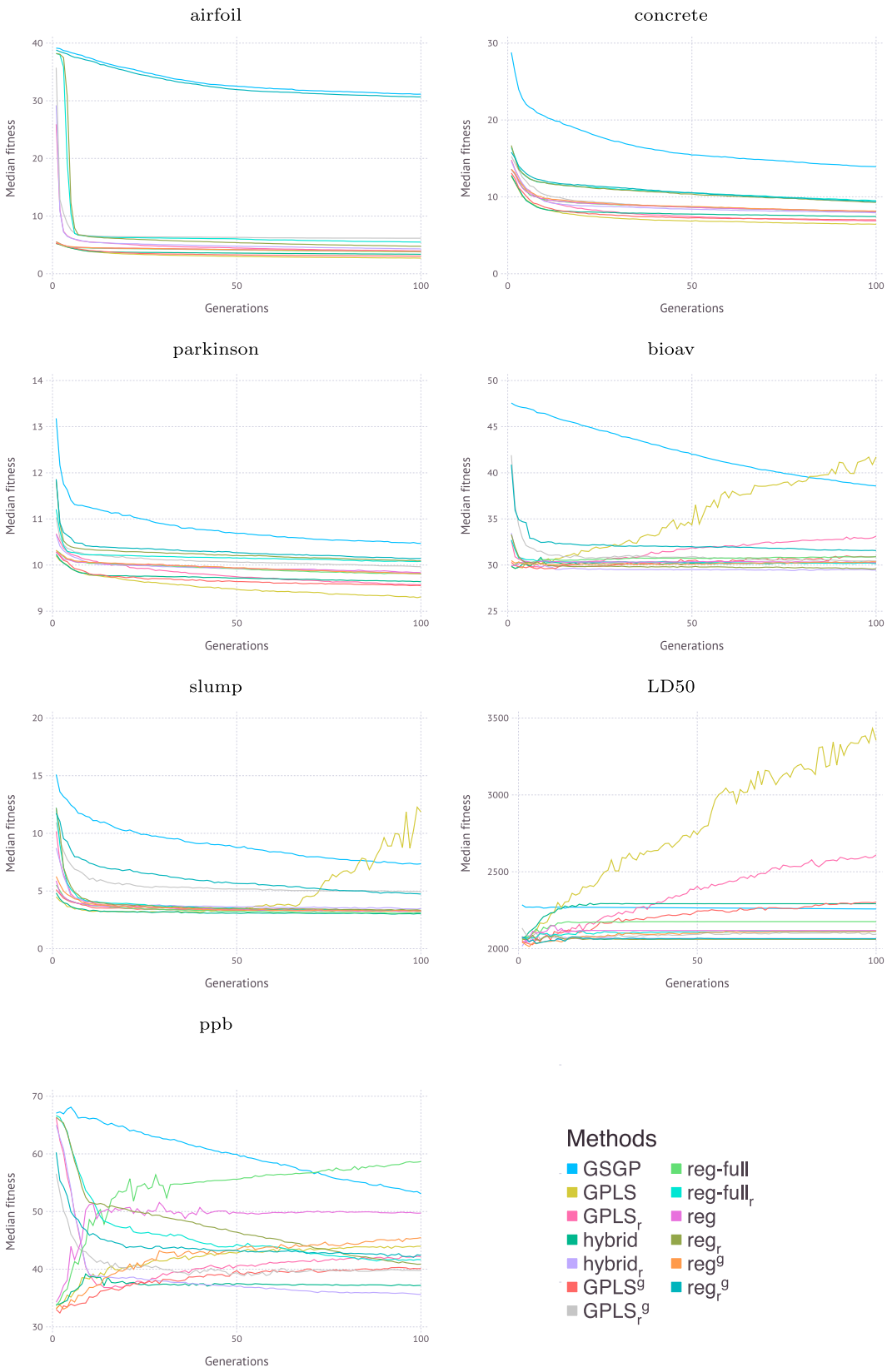

Fig. 2 Evolution of the median best fitness computed on the test set across 100 generations for all considered problems

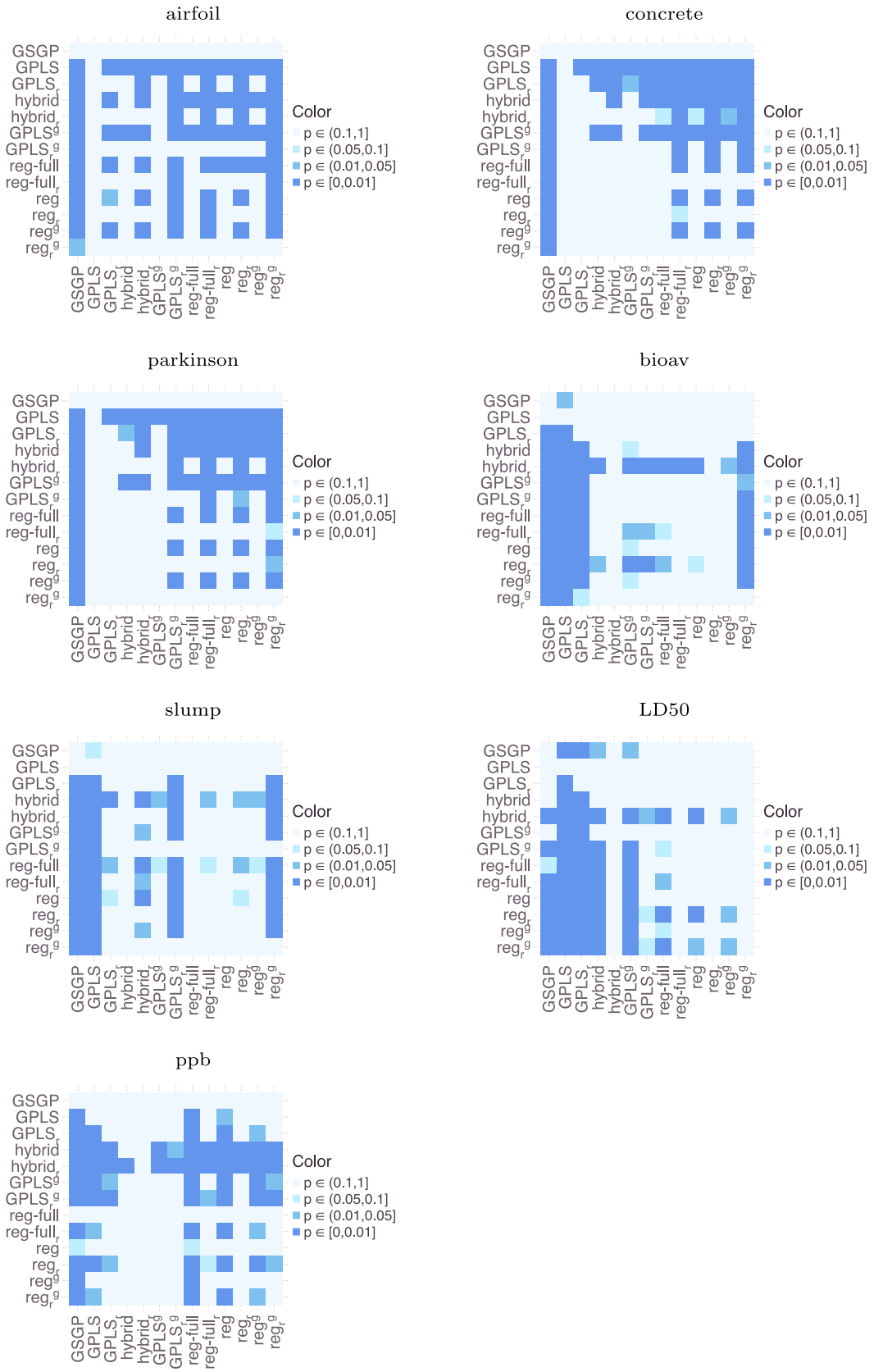

Fig. 3 Statistical comparison of the tested algorithms via a one-tailed Mann-Whitney U-test. The alternative hypothesis is that the first compared algorithm (in the rows) produces results with better fitness than the second algorithm (in the columns) more than 1/2 of the times

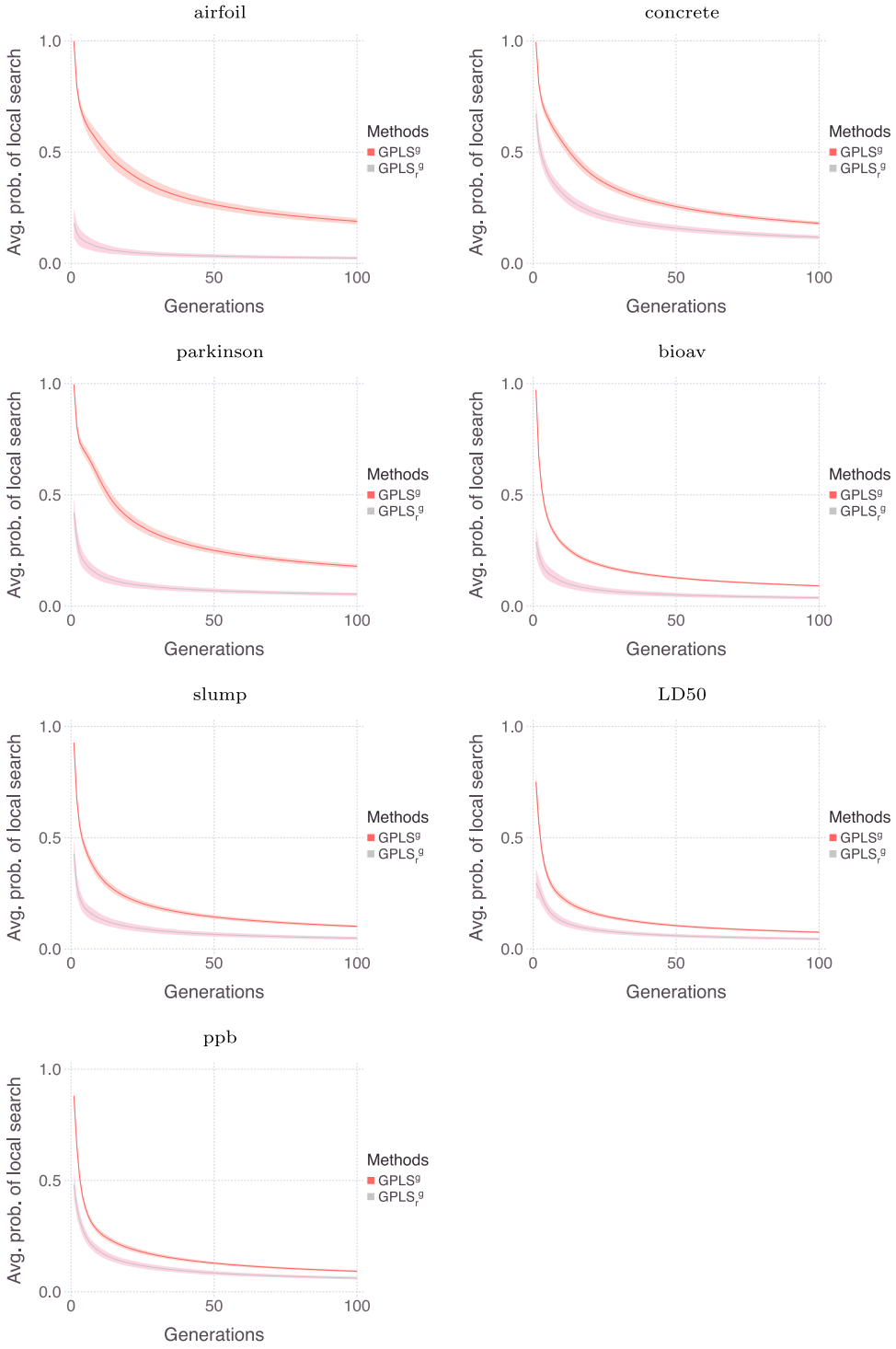

Fig. 4 The evolution of the average probability of applying the local search for all methods using the gen algorithm where local search can be applied for all generations. The shadow represents one standard deviation above and below the average

5 Results

The distribution of fitness values on the test set at the final generation is shown in Fig. 1. The evolution of the median best fitness, also on the test set, computed across all runs as a function of the number of generations, is reported in Fig. 2. A statistical comparison of the test set fitness values obtained at the final generation is presented in Fig. 3. To assess whether an algorithm performs significantly differently from others at a significance level of $\alpha = 0.01$, one can examine the corresponding row in the matrix and identify cells with the darkest color, indicating a p-value of at most 0.01. Given two algorithms i and j, a dark blue in position (i, j) means that algorithm i outperforms algorithm j in

a statistically significant manner. Conversely, a dark blue color in position (j, i) signifies that algorithm j significantly outperforms algorithm i. If neither position contains a dark blue cell, there is no statistically significant difference between the two algorithms' performances. Finally, Fig. 4 displays, for all methods making use of the gen algorithm, the probability of performing a local search step as a function of the number of generations.

By analyzing the results, we can identify three main behaviours:

- There is no overfitting, GPLS is the best performer, and GSGP is the worst one. This happens for the airfoil, concrete, and parkinson datasets.
- There is overfitting, GPLS is now the worst performer, and most of the other methods can manage overfitting while still outperforming GSGP. This is the case for the bioav, slump, and LD50 datasets.
- The last behaviour appears only in the ppb datasets and shows GPLS not overfitting but with other methods being the best performers. In this case, some of the competitors overfit.

From Fig. 2, it is possible to observe a notable difference in the *initial* median fitness across methods. In particular, the basic GSGP method (without any local search) consistently exhibits a lower initial median fitness compared to the other variants. At first glance, this discrepancy may appear unexpected, as all methods begin with randomly initialized populations and would therefore be expected to start with similar fitness distributions. However, it is important to note that the median fitness is plotted *after* the first generation, thereby excluding the initial random population from the visualization. This suggests that incorporating local search operators into GSGP introduces substantial improvements in solution quality as early as the first generation.

We will discuss the results on each dataset according to the above behaviours.

As it is possible to observe, for the airfoil problem, all of the algorithms produce better results than GSGP, except for reg_r^g. While there are variations among the different methods, it is possible to observe that all of them converge rapidly—in less than 10 generations—to a fitness value below 10 without any noticeable improvement after. The boxplots and the statistical tests show that GPLS is the best performer. The overfitting prevention techniques do not provide any advantage here, and, in a limited part, the RMSE tends to be slightly higher than GPLS for them. However, it is important to remark that, excluding the case of reg_r^g, all methods are still able to outperform GSGP in a statistically significant way.

For the concrete dataset, the behaviour is similar to the one on airfoil, with GPLS being the best performer, also in a statistically significant way. It is possible to observe that the reg-based methods perform worse than the ones using GSM-LS, and that ridge regression reduces the ability to find solutions with a better fitness. As with airfoil, there is a quick improvement at the beginning for most algorithms excluding GSGP, followed in some cases by a slow improvement of the fitness.

A similar but less pronounced pattern appears from the results on the parkinson dataset. GPLS is still the best performer, with GSGP being the worst one. The difference here is less pronounced, but the main characteristics found in the results on the concrete dataset are still present: the reg -based methods generally perform worse than the one based on GSM-LS, and the use of ridge regression limits the ability to find solutions with fitness on par with the methods using least squares regression. The change in fitness with time shows a fast convergence to good-quality solutions for most methods, but, differently from the airfoil and concrete datasets, some of the local search methods are still improving after 100 generations.

The bioav dataset provides a first example where GPLS is not the best performer. In this case, it is the worst-performing method and significantly worse than plain GSGP. The reg_r^g method is among the worst considering the ones that still perform better than GSGP, similar to what happened in the airfoil dataset. By observing the evolution of the median fitness over the number of generations, it is interesting to notice not only that GPLS starts to overfit after only a few generations, but also that GPLS_r is slowly getting worse and worse on the test set. This shows that, at least in this case, the use of ridge regression is not sufficient to counteract the overfitting of GPLS.

The results on the slump dataset show many similarities with the ones on the bioav dataset, with GPLS being the worst performer due to overfitting. Similarly to what we observed in other datasets, reg_r^g converges slowly to good-quality solutions compared to the other local search methods. From the evolution of the fitness with respect to the number of generations, it is possible to observe that the overfitting for GSGP starts later (after more than 50 generations) and that no overfitting seems to happen for GPLS_r, as in the case of the bioav dataset.

In the LD50 dataset, it is possible to observe overfitting for most methods based on GSM-LS, with the reg-based methods improving with respect to GSGP. An interesting pattern appears when observing the evolution of the median fitness with time. GPLS and GPLS_r immediately start to overfit, following the same pattern observed in the bioav dataset, with GPLS getting worse on the test set faster than GPLS_r. Thus, while the use of ridge regression is useful to

limit the amount of overfitting, it is not sufficient to completely counteract it, at least in the case of GPLS and when not combined with other overfitting prevention methods. An interesting remark is that overfitting also appears for GPLS^g, even though at a much slower pace.

The results on the last dataset, ppb, are different from all others since GPLS is still performing better than GSGP. However, by observing the evolution of fitness over time, it is clear that there is overfitting from the first few generations already. In general, the patterns that can be found are either a slow but constant improvement in fitness (GSGP and reg_r), an initial large improvement in the fitness followed by an almost stationary situation ($\mathsf{reg-full}_r$, reg_r^g, GPLS_r^g, and hybrid_r), and an almost immediate overfitting (all remaining methods).

If we focus only on the methods using the gen algorithm where local search can be applied for all generations (i.e., GPLS^g and GPLS_r^g) and, in particular, the probability of applying a local search step, it is possible to observe a general trend with a distinction between the use of ridge regression and the ordinary least squares regression. The general trend is a decline in probability followed by a stabilization or a slower decline. This pattern is consistent with the effect of local search in many datasets: the fitness improves rapidly at the beginning of the evolution, and, subsequently, only small improvements are possible. Hence, one can expect that successful local search steps will be fewer and fewer with time, thus leading to a general decrease in the probability of applying the local search step. Moreover, the probability decreases faster and stabilizes to smaller values when applying ridge regression. Thus, it seems that the combination of the two strategies on local search makes their contribution marginal in the evolutionary process, i.e., most local search steps are performed at the beginning.

As it is possible to observe from the results, there is no clear method outperforming all the others. Concerning the research questions that we investigated, the main conclusions that can be drawn are the following:

RQ1 Most overfitting prevention strategies are effective, and, in most cases, overfitting can be avoided. In particular, ridge regression can reduce the amount of overfitting of GPLS without eliminating it. On the other hand, the early stopping of the local search is effective in most cases but can still lead to large overfitting (e.g., in the LD50 dataset). The use of gen appears to be the more consistent in preventing overfitting, even considering the slight decrease in performance in one of the datasets (ppb for the reg^g method).

RQ2 Most overfitting prevention strategies provide consistently better performances than GSGP. Thus, even if they can hinder the effect of local search in datasets not prone to overfitting, the final effect is an increase in performance compared to GSGP and a reduction of the tendency to overfit.

RQ3 No strategy is consistently better than all the others. When there is no overfitting, plain GPLS can perform extremely well. All overfitting prevention strategies work (with caveats for some datasets, as discussed above), and can be combined, even if the combination of gen and ridge regression seems not to give the best situation.

As an additional interpretation of the results, it is possible to see that certain datasets are inherently more difficult than others. For instance, the LD50 and bioav datasets show that most methods get stuck in a local optimum quite early. In this cases, the most difficult datasets (such as LD50 and bioav) indicate that learning is probably not an appropriate approach here in general, independently of the applied method. On the other hands, for the other datasets that seem more amenable to learning, often the simplest strategy to avoid over fitting works best (such as using local search for less generations).

In summary, is it worthwhile to apply a local search step in GSGP? Yes, we can consistently obtain better fitness for some of the algorithms. Can we limit the risk of overfitting easily? Yes, there are multiple overfitting prevention strategies. Is there one preferred method to limit overfitting? While reg appears to be the most consistent one, it is not a clear winner.

6 Conclusions

In this work, we performed a comprehensive study of local search in GSGP, comparing two distinct approaches (GPLS and reg) and assessing their performance under various overfitting prevention strategies. These strategies included applying local search only during the first 10 generations, incorporating ridge regression into the local search step, and utilizing the adaptive gen algorithm.

Our results demonstrate that integrating local search can lead to performance improvements over standard GSGP. However, effective overfitting prevention is critical, as the local search variants are inherently prone to overfitting. Interestingly, no single overfitting prevention strategy consistently dominates; nonetheless, approaches based on the gen algorithm tend to yield more stable results across different settings.

In future work, it would be interesting to study the relation between overfitting and the trajectories of individuals in semantic space, to determine whether the "path" followed by an individual provides early indicators of overfitting. This analysis would help the design of better overfitting

prevention strategies. Finally, more effective local search algorithms can be defined. However, in this case, it would be necessary to consider the potential additional computational effort. While the local search methods studied here were not computationally intensive, this may not hold for more sophisticated variants.

Author contributions M.C., L.Man., and L.Mar. designed the study; L. Man. and M.C. wrote the software; L. Man. produced the figures; all the authors wrote the paper; all the authors analyzed the results; M.C. collected funding; all authors reviewed the manuscript.

Funding This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152/2025 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS - https://doi.org/10.54499/UID/04152/2025 (2025-01-01/2028-12-31) and UID/PRR/04152/2025 https://doi.org/10.54499/UID/PRR/04152/2025 (2025-01-01/ 2026-06-30).

Data availability The datasets used for this study are freely available (see citations in the manuscript). In case of problems, the readers can obtain the datasets by contacting Professor Luca Manzoni.

Declarations

Conflict of interest The authors have no financial or non-financial interests to disclose.

Ethical Approval Not applicable.

References

Aleti A, Moser I (2016) A systematic literature review of adaptive parameter control methods for evolutionary algorithms. ACM Computing Surveys (CSUR) 49(3):1–35

Archetti F, Lanzeni S, Messina E, Vanneschi L (2007) Genetic programming for computational pharmacokinetics in drug discovery and development. Genet Program Evolvable Mach 8:413–432

Azad RMA, Ryan C (2014) A simple approach to lifetime learning in genetic programming-based symbolic regression. Evol Comput 22(2):287–317

Bakurov I, Muñoz Contreras JM, Castelli M, Rodrigues N, Silva S, Trujillo L, Vanneschi L (2024) Geometric semantic genetic programming with normalized and standardized random programs. Genet Program Evolvable Mach 25(1):6

Bonin L, Rovito L, De Lorenzo A, Manzoni L (2024) Cellular geometric semantic genetic programming. Genet Program Evolvable Mach 25(1):8

Brooks TF, Pope DS, Marcolini MA (1989) Airfoil self-noise and prediction. Technical report

Castelli M, Manzoni L (2019) GSGP-C++ 2.0: A geometric semantic genetic programming framework. SoftwareX 10:100313

Castelli M, Vanneschi L, Silva S (2013) Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. Expert Syst Appl 40(17):6856–6862

Castelli M, Silva S, Vanneschi L (2015) A C++ framework for geometric semantic genetic programming. Genet Program Evolvable Mach 16(1):73–81

Castelli M, Trujillo L, Vanneschi L (2015) Energy consumption forecasting using semantic-based genetic programming with local search optimizer. Comput Intell Neurosci 2015:57

Castelli M, Trujillo L, Vanneschi L, Popovič A (2015) Prediction of energy performance of residential buildings: A genetic programming approach. Energy and Buildings 102:67–74

Castelli M, Vanneschi L, Manzoni L, Popovič A (2016) Semantic genetic programming for fast and accurate data knowledge discovery. Swarm Evol Comput 26:1–7

Castelli M, Manzoni L, Vanneschi L, Silva S, Popovič A (2016) Self-tuning geometric semantic genetic programming. Genet Program Evolvable Mach 17:55–74

Castelli M, Gonçalves I, Trujillo L, Popovič A (2017) An evolutionary system for ozone concentration forecasting. Inf Syst Front 19:1123–1132

Castelli M, Sormani R, Trujillo L, Popovič A (2017) Predicting per capita violent crimes in urban areas: an artificial intelligence approach. J Ambient Intell Humaniz Comput 8:29–36

Castelli M, Manzoni L, Mariot L, Saletta M (2019) Extending local search in geometric semantic genetic programming. In: Progress in Artificial Intelligence: 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3–6, 2019, Proceedings, Part I 19, pp. 775–787. Springer

Castelli M, Trujillo L, Vanneschi L, Silva S, et al (2015) Geometric semantic genetic programming with local search. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 999–1006. ACM

Chen X, Ong Y-S, Lim M-H, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591–607

Corus D, Oliveto PS (2024) On the generalisation performance of geometric semantic genetic programming for boolean functions: Learning block mutations. ACM Transactions on Evolutionary Learning 4(4):1–33

Črepinšek M, Liu S-H, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: A survey. ACM computing surveys (CSUR) 45(3):1–33

Dick G (2024) An ensemble learning interpretation of geometric semantic genetic programming. Genet Program Evolvable Mach 25(1):9

Doerr B, Neumann F (2021) A survey on recent progress in the theory of evolutionary algorithms for discrete optimization. ACM Transactions on Evolutionary Learning and Optimization 1(4):1–43

Doerr B, Doerr C (2019) Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices. Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, 271–321

Doerr B, Neumann F (2019) Theory of evolutionary computation: Recent developments in discrete optimization

Eiben ÁE, Hinterding R, Michalewicz Z (2002) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141

Emigdio Z, Trujillo L, Schütze O, Legrand P et al (2014) Evaluating the effects of local search in genetic programming. EVOLVE-A

Bridge Between Probability. Set Oriented Numerics, and Evolutionary Computation V. Springer, Cham, pp 213–228
Enríquez-Zárate J, Trujillo L, Lara S, Castelli M, Emigdio Z, Muñoz L, Popovič A et al (2017) Automatic modeling of a gas turbine using genetic programming: An experimental study. Appl Soft Comput 50:212–222
Eskridge BE, Hougen DF (2004) Imitating success: A memetic crossover operator for genetic programming. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), vol. 1, pp. 809–815. IEEE
Farinati D, Bakurov I, Vanneschi L (2023) A study of dynamic populations in geometric semantic genetic programming. Inf Sci 648:119513
Gagné C, Schoenauer M, Parizeau M, Tomassini M (2006) Genetic programming, validation sets, and parsimony pressure. In: Genetic Programming: 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings 9, pp. 109–120. Springer
Gao K, Cao Z, Zhang L, Chen Z, Han Y, Pan Q (2019) A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. IEEE/CAA Journal of Automatica Sinica 6(4):904–916
Gao S, Yu Y, Wang Y, Wang J, Cheng J, Zhou M (2021) Chaotic local search-based differential evolution algorithms for optimization. IEEE Transactions on Systems Man and Cybernetics Systems 51(6):3954–3967
Gonçalves I, Silva S (2013) Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings 16, pp. 73–84. Springer
Gonçalves I, Silva S, Fonseca CM (2015) On the generalization ability of geometric semantic genetic programming. In: Genetic Programming: 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 18, pp. 41–52. Springer
Gregor M, Spalek J (2016) Fitness-based adaptive control of parameters in genetic programming: adaptive value setting of mutation rate and flood mechanisms. arXiv preprint arXiv:1605.01514
Hajek P, Henriques R, Castelli M, Vanneschi L (2019) Forecasting performance of regional innovation systems using semantic-based genetic programming with local search optimizer. Computers & Operations Research 106:179–190
Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
Koza JR (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4:87–112
Koza JR (2010) Human-competitive results produced by genetic programming. Genet Program Evolvable Mach 11:251–284
Krawiec K, Pawlak T (2013) Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. Genet Program Evolvable Mach 14:31–63
Krawiec K, Lichocki P (2009) Approximating geometric crossover in semantic space. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 987–994. ACM
Lissovoi A, Oliveto PS (2019) Computational complexity analysis of genetic programming. In: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 475–518. Springer
Marquardt DW, Snee RD (1975) Ridge regression in practice. Am Stat 29(1):3–20
McDermott J, Agapitos A, Brabazon A, O'Neill M (2014) Geometric semantic genetic programming for financial data. In: European Conference on the Applications of Evolutionary Computation, pp. 215–226. Springer
McDermott J, White DR, Luke S, Manzoni L, Castelli M, Vanneschi L, Jaskowski W, Krawiec K, Harper R, De Jong K, et al (2012) Genetic programming needs better benchmarks. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, pp. 791–798
Moraglio A, Krawiec K, Johnson CG (2025) Editorial introduction to the special issue for the tenth anniversary of geometric semantic genetic programming. Genet Program Evolvable Mach 26(1):16
Moraglio A, Krawiec K, Johnson CG (2012) Geometric semantic genetic programming. In: International Conference on Parallel Problem Solving from Nature, pp. 21–31. Springer
Moraglio A, Poli R (2004) Topological interpretation of crossover. In: Genetic and Evolutionary Computation Conference, pp. 1377–1388. Springer
Moscato P et al (1989) (1989) On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report 826:37
Neri F, Cotta C (2012) Memetic algorithms and memetic computing optimization: A literature review. Swarm Evol Comput 2:1–14
Nievergelt Y (1994) Total least squares: State-of-the-art regression in numerical analysis. SIAM Rev 36(2):258–264
Papa JP, Rosa GH, Papa LP (2017) A binary-constrained geometric semantic genetic programming for feature selection purposes. Pattern Recogn Lett 100:59–66
Pietropolli G, Manzoni L, Paoletti A, Castelli M (2022) Combining geometric semantic gp with gradient-descent optimization. In: Genetic Programming: 25th European Conference, EuroGP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings, pp. 19–33. Springer
Rivero D, Fernandez-Blanco E, Fernandez-Lozano C, Pazos A (2020) Population subset selection for the use of a validation dataset for overfitting control in genetic programming. Journal of Experimental & Theoretical Artificial Intelligence 32(2):243–271
Rost A, Petrova I, Buzdalova A (2016) Adaptive parameter selection in evolutionary algorithms by reinforcement learning with dynamic discretization of parameter range. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, pp. 141–142
Russo M (2020) A novel technique to self-adapt parameters in parallel/distributed genetic programming. Soft Comput 24(22):16885–16894
Topchy A, Punch WF (2001) Faster genetic programming based on local gradient search of numeric leaf values. In: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, pp. 155–162. Morgan Kaufmann Publishers Inc
Trujillo L, Contreras JMM, Hernandez DE, Castelli M, Tapia JJ (2022) Gsgp-cuda a cuda framework for geometric semantic genetic programming. SoftwareX 18:101085
Trujillo L, Emigdio Z, Juárez-Smith PS, Legrand P, Silva S, Castelli M, Vanneschi L, Schütze O, Muñoz L, et al (2018) Local search is underused in genetic programming. In: Genetic Programming Theory and Practice XIV, pp. 119–137. Springer, Cham
Tsanas A, Little M, McSharry P, Ramig L (2009) Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests. Nature Precedings, 1–1
Vanneschi L, Castelli M, Silva S (2014) A survey of semantic methods in genetic programming. Genet Program Evolvable Mach 15(2):195–214
Vanneschi L, Castelli M, Manzoni L, Silva S (2013) A new implementation of geometric semantic gp and its application to problems in pharmacokinetics. In: European Conference on Genetic Programming, pp. 205–216. Springer
Vanneschi L, Farinati D, Rasteiro D, Rosenfeld L, Pietropolli G, Silva S (2025) Exploring non-bloating geometric semantic genetic programming. Genetic Programming Theory and Practice XXI, 237–258

White DR, McDermott J, Castelli M, Manzoni L, Goldman BW, Kronberger G, Jaśkowski W, O'Reilly U-M, Luke S (2013) Better gp benchmarks: community survey results and proposals. Genet Program Evolvable Mach 14:3–29

Yeh I-C (1998) Modeling of strength of high-performance concrete using artificial neural networks. Cem Concr Res 28(12):1797–1808

Žegklitz J, Pošík P (2015) Model selection and overfitting in genetic programming: Empirical study. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1527–1528

Zhang H, Chen Q, Xue B, Banzhaf W, Zhang M (2024) A geometric semantic macro-crossover operator for evolutionary feature construction in regression. Genet Program Evolvable Mach 25(1):2

Zhang M, Smart W (2004) Genetic programming with gradient descent search for multiclass object classification. In: European Conference on Genetic Programming, pp. 399–408. Springer