NOVA IMS Information Management School

MDSAA

Master Degree Program in

**Data Science and Advanced Analytics**

# EXPLORING CROSSOVER OPERATORS IN SLIM_GSGP: TOWARDS MORE COMPACT AND GENERALIZABLE MODELS

A multi-phase evaluation of novel crossover operators for efficient and interpretable evolutionary learning

Sofia Carreira Da Conceição Alves Pereira

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

**NOVA Information Management School**
**Instituto Superior de Estatística e Gestão de Informação**
Universidade Nova de Lisboa

**NOVA Information Management School**

**Instituto Superior de Estatística e Gestão de Informação**

Universidade NOVA de Lisboa

# EXPLORING CROSSOVER OPERATORS IN SLIM_GSGP: TOWARDS MORE COMPACT AND GENERALIZABLE MODELS

by

Sofia Carreira Da Conceição Alves Pereira

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Data Science

**Supervised by**:

Leonardo Vanneschi, Full Professor, NOVA Information Mannagment School
Davide Farinati, PhD, NOVA Information Mannagment School

July, 2025

**STATEMENT OF INTEGRITY**

I, Sofia C. Pereira, hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledge the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 14 of July 2025

**Exploring Crossover Operators in SLIM_GSGP: Towards More Compact and Generalizable Models**

**A multi-phase evaluation of novel crossover operators for efficient and interpretable evolutionary learning**



This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.3.9) (Lourenço, 2021).

# Acknowledgements

First and foremost, I am especially grateful to my thesis advisor, Professor Leonardo Vanneschi, who was the driving force behind this work. He believed in me even during moments of doubt and instilled in me the confidence to pursue paths I had not previously considered attainable. Through his guidance, I learned that the most important thing is to give one's best effort without fear. This lesson profoundly transformed the young engineering student I was upon entering this faculty, showing me that it is not only acceptable but essential to explore beyond convention — to wonder and to *dream*.

Equally, I would like to express my deepest gratitude to NOVA IMS, an institution that has demonstrated to me that the only true limitation a person faces is themselves. Its commitment to continuous improvement, coupled with a supportive and nurturing environment full of opportunities, has played a fundamental role in shaping the person I am today.

I would also like to thank all those who supported me throughout the development of this thesis. You may not realize the extent of your contribution, but without your help, this achievement would not have been possible.

Finally, I am deeply thankful to my family and friends, whose unwavering support and encouragement have sustained me and inspired the dreamer in me to keep *dreaming*.

”

*“Toutes les grandes personnes ont d’abord été des enfants. Mais peu d’entre elles s’en souviennent”*

**— Antoine de Saint-Exupéry**, Le Petit Prince

# Abstract

Genetic Programming (GP) is renowned for its ability to evolve symbolic, human-interpretable models. SLIM_GSGP (Semantic Learning with Inflate and deflate Mutations), a recent non-bloating variant of Geometric Semantic Genetic Programming (GSGP) has shown strong potential in balancing accuracy with interpretability, while maintaining GSGP's property of inducing a unimodal error surface on any supervised learning problem. Despite its promising performance, SLIM_GSGP remains under-explored in the literature. Most existing studies have focused on its mutation-based design, leaving the role and potential impact of crossover, and genetic exchange, largely unexamined. As a result, the contribution of crossover to model evolution in this context is still not well understood. This thesis investigates the role of crossover in SLIM_GSGP by proposing and evaluating a diverse set of 24 crossover operators, designed to explore both syntactic and semantic characteristics. Using a three-phase experimental methodology, we assess these operators across predictive performance, model size, and behavioral diversity. Our analysis includes Pareto-based ranking, complexity profiling, and benchmarking against mutation-only SLIM_GSGP configurations. These findings challenge the prevailing notion that crossover plays a minor role in GSGP, suggesting that, when properly defined, crossover can still play a significant role in the evolutionary process. Several of the proposed operators frequently yield models that are not only more compact, but also perform as well as or better than existing approaches in terms of generalization to unseen data. By analyzing the mechanisms behind their success, this work contributes to a deeper understanding of how crossover can be harnessed to improve the effectiveness of SLIM_GSGP, transforming it from a traditionally sidelined operator into a strategically crafted mechanism capable of fostering the evolution of symbolic models that are not only accurate and compact but also interpretable.

**Keywords:** Genetic Programming, SLIM_GSGP, GSGP, Crossover Operators, Symbolic Regression, Interpretability

## Sustainable Development Goals (SDG):

# Resumo

A Programação Genética (GP) é reconhecida pela sua capacidade de evoluir modelos simbólicos e interpretáveis por humanos. O SLIM_GSGP *(Semantic Learning with Inflate and deflate Mutations)*, uma variante *non-bloating* do *Geometric Semantic Genetic Programming (GSGP)*, demonstra um forte potencial na conciliação entre precisão e interpretabilidade, mantendo a propriedade do GSGP de induzir uma *unimodal error surface* em problemas preditivos. Apesar do seu potencial o SLIM_GSGP continua pouco explorado na literatura. A maioria dos estudos existentes tem-se focado no seu design baseado em mutações, deixando o papel e o impacto do *crossover*, e da troca genética, amplamente por investigar. Como consequência, a contribuição do crossover para a evolução dos modelos neste contexto ainda não é bem compreendida. Esta tese investiga o papel do crossover no SLIM_GSGP, propondo e avaliando um conjunto de 24 operadores, concebidos para explorar características sintáticas e semânticas. Através de uma metodologia experimental de três fases, estes operadores são avaliados em termos de desempenho preditivo, tamanho dos modelos e diversidade comportamental. A análise inclui a avaliação de *Pareto Fronts*, estudos de complexidade e comparação com o SLIM_GSGP apenas com mutação. Os resultados desafiam a noção de que o *crossover* tem um papel secundário no GSGP, sugerindo que, quando devidamente definido, o *crossover* pode desempenhar um papel significativo no processo evolutivo. Vários dos novos operadores são capazes de gerar modelos não só mais compactos, mas também com igual ou melhor capacidade de generalização para dados não vistos. Ao analisar os mecanismos subjacentes ao seu sucesso, este trabalho contribui para uma compreensão mais profunda de como o crossover pode ser aproveitado para melhorar a eficácia do SLIM_GSGP, transformando-o de um operador tradicionalmente marginalizado, num mecanismo estrategicamente concebido, capaz de fomentar a evolução de modelos simbólicos que sejam simultaneamente precisos, compactos e interpretáveis.

**Palavras-chave:** Programação Genética, SLIM_GSGP, GSGP, Crossover Operators, Regressão Simbólica, Interpretabilidade

# Contents

# List of Figures

# List of Algorithms

# List of Acronyms and Abbreviations

| | |
|---|---|
| **BBXO** | Best Block Crossover |
| **BDBC** | Best Donor Broadcast Crossover |
| **BDDC** | Distributed Best Donor Crossover |
| **BIDC** | Impact-Driven Best Donor Crossover |
| **DBC** | Donor Broadcast Crossover |
| **DDC** | Distrbuted Donor Crossover |
| **DGSM** | Deflate Mutation |
| **EA** | Evolutionary Algorithms |
| **GAs** | Genetic Algorithms |
| **GC** | Geometric Crossover |
| **GM** | Geometric Mutation |
| **GP** | Genetic Programming |
| **GSC** | Geometric Semantic Crossover |
| **GSGP** | Geometric Semantic Genetic Programming |
| **GSM** | Geometric Semantic Mutation |
| **GSO** | Geometric Semantic Operators |
| **IDC** | Impact-Driven Donor Crossover |
| **IGSM** | Inflate Mutation |
| **ML** | Machine Learning |
| **RBC** | Random Block Crosovers |
| **RMSE** | Root Mean Squared Error |
| **SBC** | Strategic Block Crosovers |

**SLIM_GSGP** Semantic Learning algorithm with Inflate and deflate Mutations

**XOBDn** Best Donor Crossover
**XODn** Donor Crossover

# 1

# Introduction

In recent years, the development of interpretable and efficient machine learning models has become increasingly important, particularly in domains where model transparency and computational scalability are essential. Genetic Programming (GP), first introduced in (Koza, 1992), has emerged as a promising paradigm due to its inherent flexibility and its ability to solve a wide range of problems across various domains. This paradigm introduced the foundational concept that *"fitness begets structure"*, which remains central to the field of GP today. Despite its advantages, traditional GP is hindered by significant drawbacks, most notably, its reliance on a blind exploration of the syntactic space, without consideration for the distribution of individuals in the semantic space (the vector of output values of a program over a set of observations).

The need for a precise correspondence in the semantic space between the actions performed on the syntax and their semantic effect led to the introduction of Geometric Semantic Genetic Programming (GSGP) by (Moraglio et al., 2012), who proposed a new class of genetic operators called Geometric Semantic Operators (GSO). These operators act directly in the semantic space, enabling more stable and predictable search dynamics by ensuring consistent effects on the semantic properties of individuals. Notably, GSO produce unimodal error surfaces, fitness landscapes devoid of local optima, which provide GSGP with a significant advantage in terms of evolvability over traditional approaches.

However, GSGP also introduces new challenges. Although GSO offer considerable benefits, their properties inevitably lead to offspring that inherit the complete structure of their parents along with additional complementary components. This results in a very fast growth in individual size which leads to increasingly unmanageable population sizes. Consequently, models become overly complex and difficult to interpret, making GSGP impractical for many real-world applications unless growth is explicitly controlled.

In response to these challenges, Semantic Learning algorithm with Inflate and deflate Mutations (SLIM_GSGP) was proposed by (Vanneschi, 2024). This framework

incorporates a novel mutation strategy, the deflate mutation, that allows for model simplification during the evolutionary process. SLIM_GSGP maintains the theoretical advantages of GSGP while achieving more compact models and, in some cases, better generalization. In fact, it has been shown to often have equal or better performance than both standard GSGP and traditional GP on unseen data, producing substantially smaller and more interpretable models than GSGP. Moreover, since SLIM_GSGP represents genotypes as linked lists, the Geometric Semantic Mutation (GSM), renamed inflate mutation, can be implemented as a simple list append, while the deflate mutation corresponds to a deletion operation. These improvements make SLIM_GSGP a promising direction for semantic genetic programming in practical applications.

Yet, a key component of the evolutionary process remains underexplored in this context: crossover operators. While the Geometric Semantic Crossover (GSC) in traditional GSGP has demonstrated limited effectiveness, often contributing little to performance and exacerbating bloating (Castelli et al., 2016), the unique structural characteristics of SLIM_GSGP open new possibilities for revisiting and redesigning crossover mechanisms. SLIM's use of linked-list representations and its inherent support for structural simplification create an opportunity to reintroduce crossover in a more controlled and meaningful way. In this context, crossover could serve not only as a source of semantic diversity but also as a mechanism for producing smaller, more interpretable models. Exploring this potential represents a promising and timely direction for advancing the capabilities of semantic genetic programming.

This thesis explores the role and design of crossover operators within the SLIM_GSGP framework, with the goal of understanding whether new or adapted crossover mechanisms can enhance model quality, reduce structural complexity, and positively influence the evolutionary dynamics of the algorithm. In particular, the study investigates:

- How do different crossover designs affect model performance and size in SLIM_GSGP?
- Can crossover be reintroduced as a meaningful component in SLIM_GSGP through novel operator designs aligned with its architecture?

To address these questions, this work proposes and evaluates several novel crossover mechanisms,including random block exchange, donor-based exchange, and block selection strategies. These operators are specifically designed to align with SLIM's internal representation of individuals as linked lists, enabling genetic material to be exchanged in a semantically meaningful and size-conscious manner. The evaluation is conducted using two of the best-performing SLIM_GSGP variants previously identified in the literature (Vanneschi et al., 2025), *SLIM+2SIG* (additive) and *SLIM*ABS* (multiplicative). Each crossover design is embedded into these configurations and tested across six symbolic regression benchmark datasets commonly used in genetic programming research.

To assess the effectiveness of each crossover, the experimental study was organized into three structured phases. **Phase 1** involved ranking and Pareto-based evaluation: 24 crossover configurations were assessed on test RMSE and model size, with top performers identified using Mann-Whitney U tests. When no clear winner emerged, a multi-objective selection using NSGA-II highlighted crossovers that best balanced performance and interpretability. **Phase 2** focused on behavioral analysis and complexity assessment, examining the evolutionary dynamics of selected crossovers through fitness and size trajectories across generations, and measuring their computational cost via average complexity metrics. This phase ensured that selected operators were not only accurate, but also efficient and stable. Finally, **Phase 3** benchmarked the best crossover-enhanced variants against the mutation-only SLIM_GSGP. Statistical comparisons and visual analyses determined whether crossover offered a significant and consistent improvement. This three-phase methodology provided a rigorous, multi-perspective evaluation of crossover integration in SLIM_GSGP, highlighting both its theoretical significance and practical value.

The remainder of this thesis is structured as follows: Chapter 2, Theoretical Background, introduces the fundamental concepts required to contextualize this work, including an overview of Machine Learning and Evolutionary Algorithms. Chapter 3, Literature Review, presents previous research relevant to the study, with a particular focus on the use of semantics and crossover in Genetic Programming. Chapter 4, Methodology, describes the SLIM_GSGP algorithm and the proposed crossover mechanisms and experimental design, including parameter settings and the statistical procedures used for evaluation. The experimental findings and a detailed discussion and interpretation of the results are reported in Chapter 5. Finally, Chapter 6 summarizes the main contributions of this work, outlines its limitations, and proposes directions for future research.

# 2

# Theoretical Background

This chapter introduces the foundational concepts necessary for understanding the work developed in this thesis. Section 2.1 provides an overview of the core principles of Machine Learning (ML), with a particular focus on supervised learning and symbolic regression. Section 2.2 explores the class of Evolutionary Algorithms (EAs), emphasizing the principles of Genetic Programming (GP), its semantic variant Geometric Semantic Genetic Programming (GSGP), and the SLIM_GSGP framework used in this study.

## 2.1 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence concerned with developing algorithms that allow computers to learn from data, identify patterns, and make decisions or predictions without being explicitly programmed. The central concept is that learning can occur by exposure to data and iterative refinement of predictions based on performance, mimicking the way humans improve at tasks through experience and feedback.

The question *"Can machines learn?"* has been central to computational thinking since Alan Turing's pioneering work on machine intelligence (Turning, 1950). Decades later, Tom Mitchell provided a widely accepted formal definition of machine learning: *"A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$"* (Mitchell, 1997). In essence, this definition captures the idea of learning as the process of *improving through experience*.

Machine Learning encompasses a wide range of techniques that allow systems to learn patterns from data. A fundamental way to categorize ML approaches is based on whether the learning process uses labeled data. This leads to a primary distinction between **supervised learning** and **unsupervised learning**.

In **supervised learning**, the algorithm is trained on a dataset where each example includes both input features and a corresponding output label. The goal is to learn a mapping from inputs to outputs, enabling the model to make accurate predictions on new, unseen data. Supervised learning is commonly used for:

- **Classification**, where the target is a discrete category (e.g., diagnosing whether a tumor is benign or malignant).
- **Regression**, where the target is a continuous value (e.g., predicting blood pressure or stock prices).

In contrast, **unsupervised learning** is applied to data without labeled outcomes. Instead of predicting a target, the algorithm seeks to discover hidden patterns or structures within the data. Typical unsupervised tasks include:

- **Clustering**, which groups similar data points based on feature similarity (e.g., segmenting patients into risk groups).
- **Dimensionality Reduction**, which projects data into a lower-dimensional space while preserving its essential structure (e.g., for visualization or noise reduction).

A third category, known as **semi-supervised learning**, combines both labeled and unlabeled data. It is particularly useful when obtaining labels is costly or time-consuming, allowing models to benefit from the large amount of available unlabeled data while still leveraging the guidance of labeled examples.

This thesis focuses on **supervised learning**, and more specifically, on **symbolic regression**, the task of learning a mathematical expression that models the underlying data. Unlike traditional regression methods, which assume a fixed functional form (e.g., linear or polynomial) and optimize only the parameters, symbolic regression simultaneously searches for both the structure and the parameters of the predictive function. For example, instead of fitting a model of the form $y = ax + b$, symbolic regression might discover an entirely different expression such as $y = \cos(x^4) + \exp(2x + 5)$, if it better fits the data.

A key advantage of symbolic regression is its inherent **interpretability**. Because the output is a human-readable mathematical expression, the resulting models are often more transparent and easier to analyze than black-box approaches like neural networks or ensemble methods. This is especially valuable in high-stakes domains such as healthcare or finance, where understanding the reasoning behind predictions is critical.

According to Vanneschi and Silva (Vanneschi & Silva, 2023), supervised learning can be framed as a functional approximation problem. Given a dataset:

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}, \tag{2.1}$$

the goal is to find or approximate a target function $\phi : \mathbb{R}^m \rightarrow \mathbb{R}$ such that:

$$\forall i = 1, 2, ..., n : \phi(\mathbf{x}_i) = y_i, \tag{2.2}$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{im})$ is an $m$-dimensional input vector, and $y_i$ is a known target value. Symbolic regression therefore operates in a high-dimensional feature space, seeking mathematical expressions that accurately model the relationship between inputs and outputs.

In order to assess the quality of a predictive model, a **loss function** or **error metric** is required. These metrics provide feedback that guides the learning process and allow for comparison between models. One commonly used error function in regression tasks is the **Root Mean Squared Error (RMSE)**, defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2.3}$$

where $y_i$ is the true target value, and $\hat{y}_i$ is the model's prediction. RMSE is sensitive to large deviations due to the squaring of errors and provides a direct measure of the average prediction error magnitude in the original units of the output variable. Its interpretability and sensitivity to variance make it a widely accepted evaluation metric in symbolic regression.

The choice of an appropriate error metric is fundamental because it provides the feedback necessary to guide the learning process. Improving model performance by minimizing such error metrics is the essence of **optimization**. However, the search for the best model often involves navigating a vast and complex space of possible solutions, where traditional optimization methods may struggle.
To address this challenge, **Evolutionary Algorithms** draw inspiration from the principles of natural selection and evolution to effectively explore and optimize within these complex search spaces. The next section highlights the foundational concepts of optimization and evolutionary algorithms, setting the stage for their application in symbolic regression.

## 2.2 Evolutionary Algorithms

Evolutionary Algorithms (EA) are a class of algorithms inspired by the principles of natural evolution, as first described by Charles Darwin in his seminal work on the *Theory of Evolution by Natural Selection* (Darwin, 1859). Darwin's insight, that species evolve over generations through the survival and reproduction of the fittest individuals, offers a powerful metaphor for solving complex computational problems through iterative improvement and adaptation.

In the early days of artificial intelligence, most research focused on emulating human intelligence, treating it as the pinnacle of evolutionary development, seeking to replicate the final product of evolution. In contrast, Lawrence Fogel proposed a paradigm shift: rather than modeling human intelligence directly, he sought to model the evolutionary process itself as a mechanism for producing intelligent behavior (Fogel et al., 1964). This idea gave rise to *Evolutionary Programming*, one of the earliest forms of evolutionary computation. Fogel viewed intelligence as the ability to predict future states of the environment and act accordingly to achieve defined goals, a process well-aligned with evolution's capacity to adapt and optimize.

Following this conceptual breakthrough, researchers continued to formalize computational analogues of evolution. In 1975, John Holland introduced **Genetic Algorithms** (GAs), which simulate evolution using populations of encoded solutions and operators such as selection, crossover, and mutation (Holland, 1975). Later, in 1992, John Koza popularized *Genetic Programming* (GP) as a versatile framework for evolving computer programs represented as syntax trees, emphasizing the idea that *"fitness begets structure"* (Koza, 1992).

Together, these pioneering works laid the foundation for Evolutionary Algorithms as a flexible and powerful approach to optimization, search, and machine learning problems. In particular, Genetic Programming has emerged as a highly expressive paradigm for evolving symbolic representations, making it especially suited for tasks such as symbolic regression. Over time, this field has seen significant advances, including the incorporation of semantic awareness and novel variation operators that aim to improve generalization and search efficiency. The following sections trace the evolution of these ideas, from the broader landscape of optimization and Genetic Algorithms to the development of Genetic Programming, its semantic extensions, and ultimately, the SLIM_GSGP framework explored in this thesis.

### 2.2.1 Optimization Problems

Optimization is a fundamental field of study focused on developing methods, strategies, and algorithms to solve complex problems by identifying the best possible solution(s) from a vast space of alternatives. At its core, an optimization problem aims to maximize or minimize a particular objective, often formalized as a mathematical function that quantifies solution quality. Formally (Vanneschi & Silva, 2023), an instance of an optimization problem is defined as a pair $(S, f)$ where:

- $S$ is the *solution space* or *search space*, representing the set of all possible candidate solutions;
- $f : S \to \mathbb{R}$ is the *fitness function* (also known as the objective or cost function), which assigns a real-valued score to each solution, quantifying its quality.

Depending on the problem, the goal may be to maximize or minimize $f$. In a *maximization problem*, higher values of $f$ are preferred, while in a *minimization problem*, lower values indicate better solutions.

The optimal solution to such a problem is called the *global optimum*. However, in practice, optimization algorithms often converge to *local optima*, solutions that are better than all others in their immediate neighborhood, yet not the best overall. While every global optimum is also a local optimum, the converse does not hold. Whether an algorithm reaches a global or local optimum depends on various factors, including the landscape of the search space and the algorithm's design.

A useful tool for analyzing the complexity of optimization problems is the concept of a *fitness landscape* (Vanneschi & Silva, 2023). Given an instance $(S, f)$ and a neighborhood structure $N$, a fitness landscape (FL) is defined by the triple $(S, f, N)$. It provides an intuitive visualization where:

- each point along the horizontal axis corresponds to a solution in $S$, ordered according to the neighborhood structure $N$;
- the vertical axis shows the fitness value $f(i)$ for each solution $i \in S$.

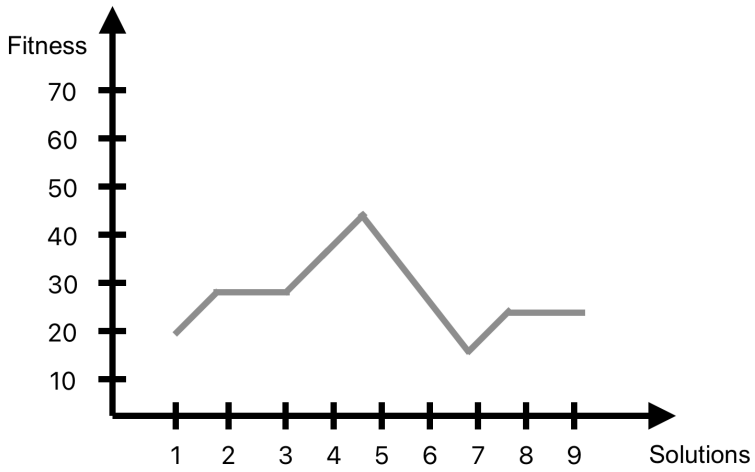


Figure 2.1: Illustration of a fitness landscape.

Fitness landscapes can be either *smooth*, with few prominent peaks, or *rugged*, with many sharp peaks and valleys. Smooth landscapes are typically easier to navigate, while rugged landscapes increase the likelihood of premature convergence to suboptimal solutions.

Another important concept is that of *unimodality*. A fitness landscape is said to be *unimodal* if it contains no local optima other than the global optimum. In such cases, most optimization algorithms are more likely to converge to a near-optimal or optimal solution, even with relatively simple strategies.

To navigate these landscapes, a variety of optimization algorithms have been proposed. *Hill Climbing*, for instance, is a straightforward local search technique that iteratively moves to a neighbor solution if it results in a fitness improvement. While effective on unimodal landscapes, it often becomes trapped in local optima in more complex, multimodal problems.

To address this, more sophisticated methods such as *Simulated Annealing* were developed. Simulated Annealing introduces a probabilistic acceptance criterion, allowing the algorithm to occasionally accept worse solutions. This stochastic element enables it to escape local optima and explore the search space more broadly. Over time, the probability of accepting worse solutions decreases, guiding the search toward convergence.

Although these techniques have proven useful, they operate on single solutions at a time and may struggle in high-dimensional or highly rugged landscapes. This limitation led to the development of population-based methods such as Genetic Algorithms, which evaluate and evolve a set of solutions simultaneously. These evolutionary approaches are particularly well-suited for complex search and optimization problems and are discussed in the following section.

### 2.2.2 Genetic Algorithms (GAs)

To address the limitations of single-solution methods like Hill Climbing and Simulated Annealing, **Genetic Algorithms (GAs)** were introduced by John Holland (Holland, 1975) as a population-based optimization technique inspired by Darwinian evolution (Darwin, 1859). Unlike the previously mentioned methods that operate on a single solution, GAs maintain a *population* of candidate solutions, referred to as *individuals*, and evolve them over successive iterations called *generations*.

Individuals in GAs are commonly represented as strings of fixed length, often binary or symbolic, that encode potential solutions. The evolutionary process is driven by five biological principles (Vanneschi & Silva, 2023), which guide not only the design of Genetic Algorithms but also future developments like Genetic Programming (GP) and Geometric Semantic Genetic Programming (GSGP):

- **Reproduction**: Individuals (parents) produce offspring, ensuring the continuity of the population.
- **Adaptation**: The fitness of individuals reflects their ability to survive and reproduce within a specific environment.
- **Inheritance**: Offspring inherit traits from their parents, preserving valuable genetic information.

- **Variation**: Offspring differ from their parents, introducing novelty and diversity.
- **Competition**: Limited resources force individuals to compete for survival, favoring those better adapted.

These principles are translated into the following core pipeline of a Genetic Algorithm, illustrated in Figure 2.2:

1. **Selection Process**: A subset $P' \subseteq P$ of size $N'$ is selected from the current population $P$ (of size $N$) based on fitness.
2. **Variation**: Selected individuals are modified via genetic operators, *crossover* and *mutation*, producing a new set $P''$ of size $N$.
3. **Survivor Selection**: A new generation is formed by selecting $N$ individuals to compose the next population $P$, restarting the cycle.

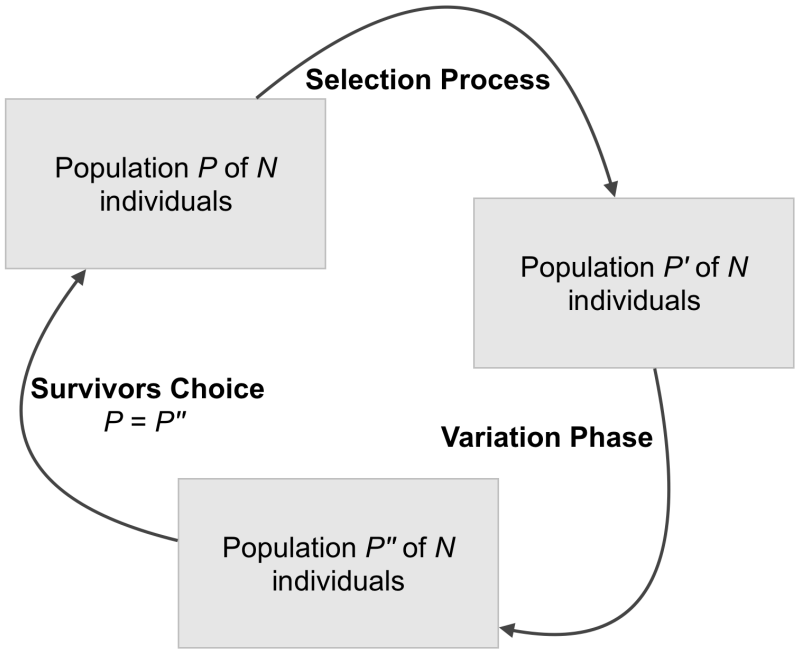


Figure 2.2: Core structure of a Genetic Algorithm: selection, variation, and survivor selection.

**Selection** plays a central role in mimicking *competition* and *adaptation*. Individuals are chosen based on fitness, with better-performing ones having a higher probability of contributing to the next generation. A widely used method, and the one adopted in this thesis, is **Tournament Selection**. In this approach, $k$ individuals are randomly sampled from the population, and the one with the best fitness is selected. This mechanism ensures both selection pressure and diversity.

Variation is responsible for introducing diversity and exploring the search space. It embodies the principles of *reproduction*, *inheritance*, and *variation* through two key genetic operators: **crossover** and **mutation**.

**Mutation** is considered the main source of innovation. It alters one or more positions (genes) in an individual's structure with a given probability $p_m$ (mutation rate). This operation allows any possible solution to eventually be generated, thus maintaining genetic diversity and preventing stagnation (Figure 2.5).

**Crossover**, by contrast, is a conservative operator that recombines information from two parent individuals. A standard one-point crossover splits both parents at a randomly chosen position and exchanges their substrings to produce two offspring. While this promotes the reuse of high-quality building blocks, it also introduces the well-known **position problem**: important structural patterns may be disrupted due to the arbitrary nature of the split (Figure 2.5).

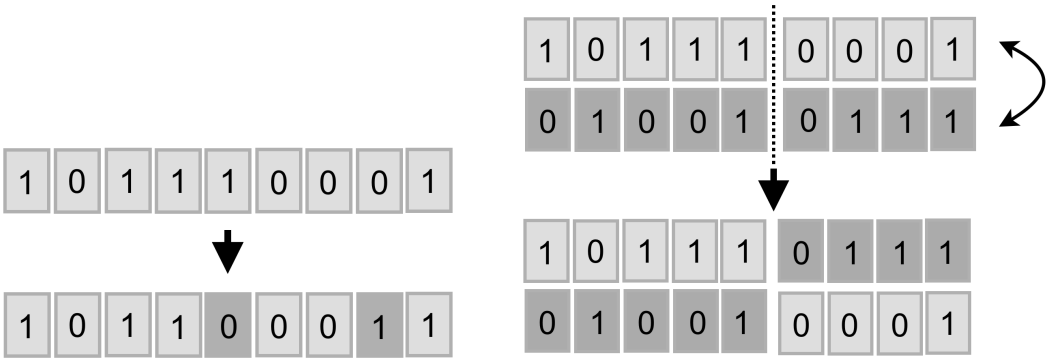


Figure 2.5: Illustration of standard genetic operators used in Genetic Algorithms: mutation (left) introduces new genetic material, and crossover (right) recombines genetic material from two parents.

To preserve the best solutions found, GAs often employ **elitism**, ensuring that the top-performing individuals are copied unaltered into the next generation.

Despite their versatility and success across numerous domains, Genetic Algorithms are not without drawbacks:

- **Premature convergence (loss of diversity)**: Populations may become too homogeneous, especially if diversity-reinforcing mechanisms are weak. This often results in convergence to local, suboptimal solutions.
- **Unicity of fitness**: GAs traditionally optimize a single fitness function, which may be limiting in real-world applications where multiple objectives must be balanced.

To mitigate this last issue, this thesis employs **Pareto Multiobjective Optimization**, which allows the simultaneous optimization of several fitness functions. A solution is considered **Pareto optimal** if no other solution is strictly better across all objectives. As illustrated in Figure 2.6, the set of non-dominated solutions forms the *Pareto front*, representing optimal trade-offs among conflicting objectives. In our context, this allows us to optimize not only for fitness but also for interpretability (e.g., solution size).

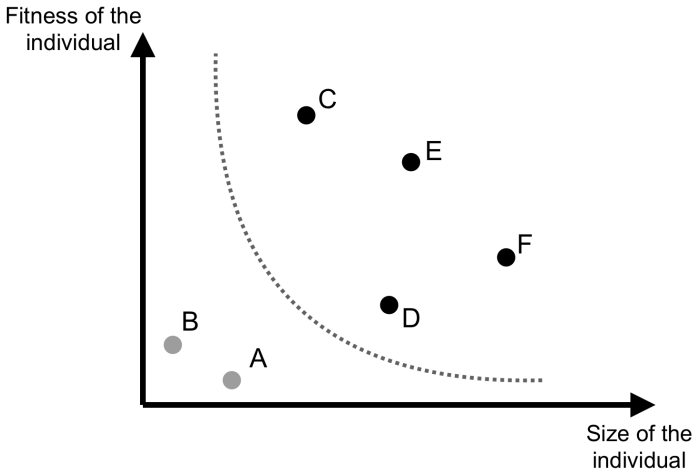


Figure 2.6: Pareto front illustrating the trade-off between two objectives (fitness and size). Points A and B are non-dominated solutions. The dashed line represents the pareto front.

### 2.2.3 Continuous Optimization and Geometric Operators

Among the various types of optimization problems, this work focuses on *continuous optimization*, where individuals are represented as vectors in an $m$-dimensional Cartesian space: $\mathbf{x} = [x_1, x_2, \ldots, x_m]$.

This representation allows the use of operators that directly exploit the geometry of the search space. One of the most influential approaches to emerge from this perspective is the concept of **geometric genetic operators**, which form the foundation of the work developed in this thesis (Moraglio et al., 2012). The interpretation of individuals as points in a continuous space enables the definition of operators that act based on geometric principles:

**Geometric Mutation (GM)** takes an individual $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ and generates a mutated offspring as follows:

$$\mathit{offspring} = [x_1 + r_1, x_2 + r_2, \ldots, x_m + r_m]$$

where each $r_i$ is drawn from a uniform distribution $[-ms, ms]$, with $ms$ representing the mutation step size.

Intuitively, GM introduces a small random perturbation to each dimension of the individual, producing a new point within a hypercube (or ball) centered around the parent. In two dimensions, this corresponds to generating offspring within a box-shaped or circular neighborhood (see Figure 2.9). This property gives rise to the name *ball mutation*. Notably, under fitness landscapes where fitness is inversely proportional to the distance to the global optimum, GM ensures that within each mutation ball, there always exists at least one point better than the parent. This leads to a *unimodal fitness landscape*, one with only a global optimum and no local optima, making the search process smoother and more efficient.

**Geometric Crossover (GC)**, in contrast, combines two parents $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ and $\mathbf{y} = [y_1, y_2, \ldots, y_m]$ to produce an offspring:

$$offspring(\mathbf{x}, \mathbf{y}) = [r_1 x_1 + (1 - r_1) y_1, r_2 x_2 + (1 - r_2) y_2, \ldots, r_m x_m + (1 - r_m) y_m]$$

where $r_i$ represnts a random number extracted with unifrom distribution from the interval [0,1], for each $i = 1, \ldots, m$.

This operator generates an offspring located on the line segment connecting the two parents in the search space. As a result, in fitness landscapes where proximity to the global optimum correlates with fitness, the offspring is guaranteed to be no worse than the worst parent. This addresses the well-known *crossover position problem* of standard genetic crossover (see Figure 2.9).

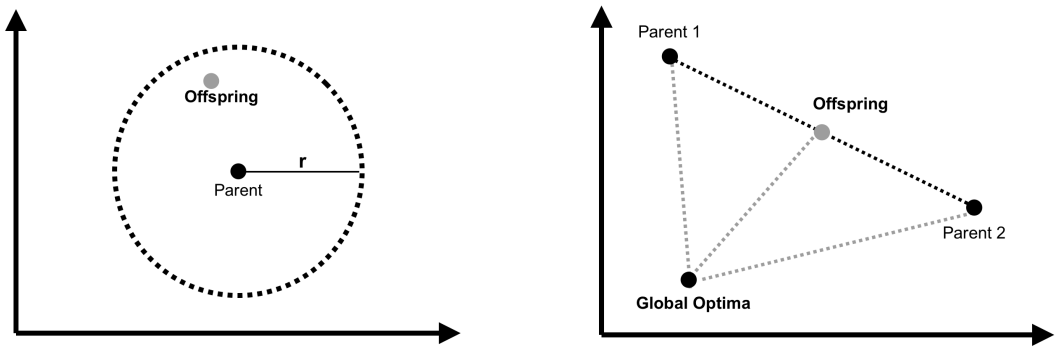


Figure 2.9: Illustration of geometric operators: (a) Geometric Mutation perturbs an individual within a defined neighborhood; (b) Geometric Crossover generates an offspring along the line segment connecting two parents.

By redefining operators in geometric terms and understanding their effects on the search space, this perspective laid the groundwork for **Geometric Semantic Genetic Programming** (GSGP), which will be studied in the following sections.

### 2.2.4 Genetic Programming (GP)

Originally introduced by Koza in (Koza, 1992), **Genetic Programming** (GP) is an evolutionary algorithm inspired by the principles of Darwinian evolution, similar to Genetic Algorithms (GAs). However, in contrast to GAs, which evolve fixed-length strings (often representing binary-encoded solutions), GP evolves computer programs directly, typically represented as syntax trees.

For example, the expression:

$$[(x_1 + x_3) * x_1] - [3 - (x_2/5)]$$

can be visualized as a tree structure, where internal nodes represent functions (e.g., $+, -, \times, \div$) and leaf nodes represent terminals (e.g., input variables or constants):

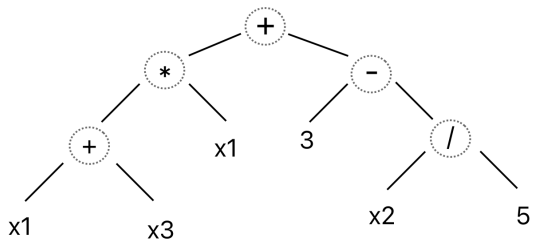


Figure 2.10: Tree-based representation of the expression $[(x_1 + x_3) * x_1] - [3 - (x_2/5)]$.

The overall evolutionary process in GP is similar to that of GAs, including selection, variation, and survival. One key difference lies in the application of variation operators: GP typically applies either mutation or crossover to each individual (not both), chosen based on predefined probabilities.

Before evolution can begin, an initial population must be generated. Several methods exist for initializing trees, the most common being the Full, Grow, and Ramped Half-and-Half (RHH) methods (Koza, 1992). This work adopts the RHH method, which aims to maximize initial population diversity by combining the properties of the Full and Grow methods:

- **Full method:** Generates trees where all branches reach a fixed maximum depth, by selecting function as nodes until the final depth (where terminals are selected).
- **Grow method:** Allows nodes to be randomly chosen as either functions or terminals, producing trees of variable shapes and sizes.
- **Ramped Half-and-Half (RHH):** Combines both methods across varying depths to ensure structural diversity in the initial population.

Because individuals are structured as trees, GP requires specific genetic operators designed to handle tree manipulations. While selection mechanisms such as tournament selection are shared with GAs, since they depend solely on fitness values, the variation operators must respect the hierarchical nature of program trees. In GP, mutation and crossover are redefined to manipulate tree structures:

**Standard Mutation:** randomly selects a subtree within the parent program and replaces it with a newly generated random subtree. This new subtree is typically generated using the same initialization method used initially, ensuring syntactic validity (Figure 2.11).

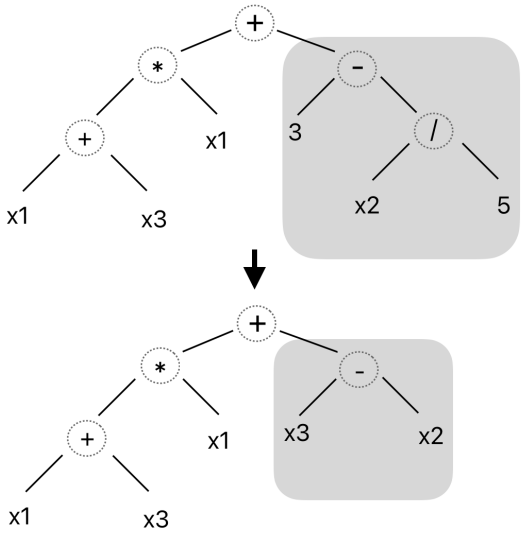


Figure 2.11: Standard GP Mutation: A randomly selected subtree (highlighted in grey) is replaced by a newly generated random subtree.

**Standard Crossover:** exchanges genetic material by selecting a random subtree in each parent and swapping them, creating new offspring that combine traits of both. Crossover helps explore new areas of the search space while maintaining viable tree structures (Figure 2.12).

Despite its success in a wide range of applications, standard GP suffers from several known issues:

- **Overfitting:** GP can produce overly complex solutions that perform well on training data but generalize poorly on unseen data.
- **Premature Convergence:** Loss of diversity during the evolutionary process may lead the population to converge prematurely to suboptimal solutions.
- **Bloat:** A common problem where individuals grow excessively large without corresponding improvements in fitness.

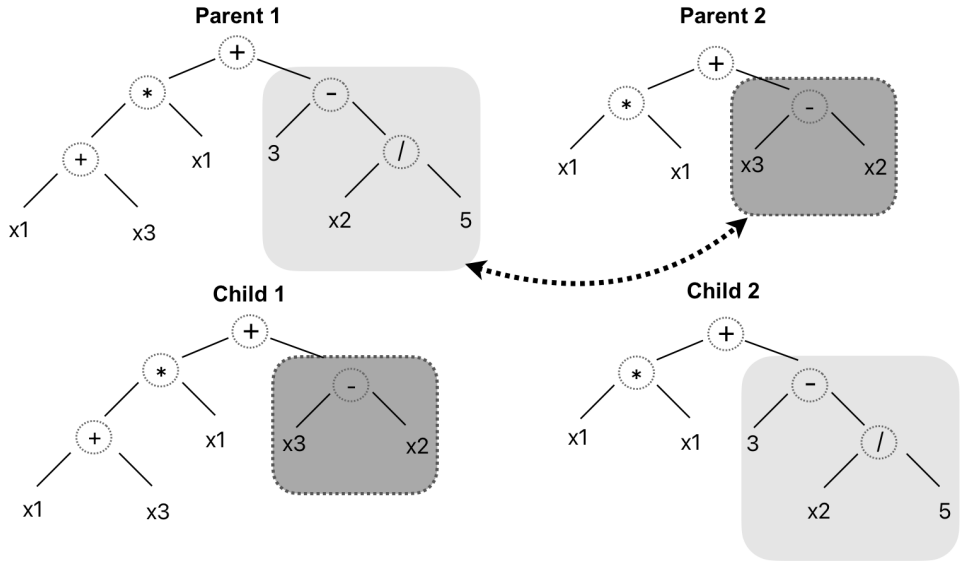


Figure 2.12: Standard GP Crossover: Two parent trees exchange randomly selected subtrees to generate offspring.

These limitations motivate the development of geometry-aware approaches to GP. One such approach is *Geometric Semantic Genetic Programming* (GSGP), which redefines genetic operators to act directly in the semantic space of programs, explained in the following section.

### 2.2.5 Geometric Semantic Genetic Programming (GSGP)

Traditional Genetic Programming (GP) operates in the syntactic space of tree-based representations, meaning genetic operators act by manipulating program structures without explicit regard for their behavior or output. This syntactic-driven search can be inefficient, as small changes in structure may have unpredictable or erratic effects on the program's output.

To overcome this limitation, *Geometric Semantic Genetic Programming* (GSGP), introduced by (Moraglio et al., 2012), proposes a different approach: evolving individuals in the **semantic space**. In GSGP, the *semantics* of an individual is defined as the vector of its output values on the training set, that is, each individual corresponds to a point in an $n$-dimensional space, where $n$ is the number of training instances (Figure 2.13).

This mapping between the genotypic (tree-based) and semantic (output-based) space allows the definition of genetic operators that act with predictable effects on an individual's behavior. Moraglio et al. (Moraglio et al., 2012) extended the concept of *Geometric Operators*, originally developed in the context of Genetic Algorithms (GAs), to the domain of Genetic Programming. In GSGP, this principle is applied to the semantic space, resulting in *Geometric Semantic Operators* (GSOs) that enable a structured and smooth search guided by output behavior, rather than relying on blind syntactic manipulation.

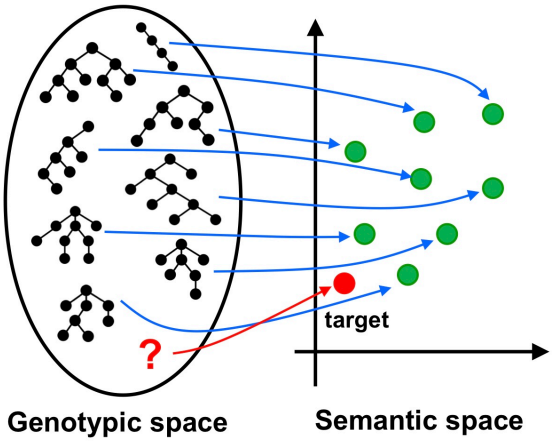


Figure 2.13: Representation of individuals in the semantic space. *Adapted from* (Vanneschi & Silva, 2023).

**Geometric Semantic Mutation (GSM)** is defined as follows. Given a parent function $T : \mathbb{R}^n \rightarrow \mathbb{R}$ and a mutation step size $ms$, GSM returns the function:

$$T_M = T + ms \cdot (T_{R1} - T_{R2})$$

where $T_{R1}$ and $T_{R2}$ are randomly generated real-valued functions. The difference $(T_{R1} - T_{R2})$ is centered around zero, so the offspring's semantics represents a weak perturbation of the parent's semantics. The magnitude of this perturbation is controlled by the scalar $ms$.

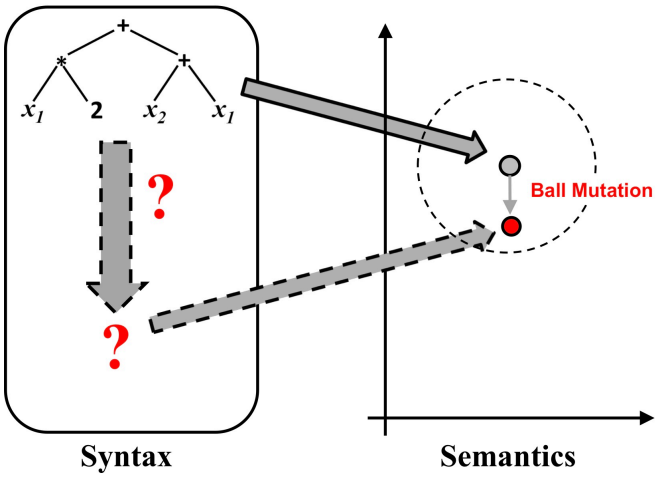


Figure 2.14: Geometric Semantic Mutation: a weak perturbation around the parent's semantic vector. *Adapted from* (Vanneschi & Silva, 2023).

This operator induces a *ball mutation* in the semantic space (Figure 2.14), which results in a *unimodal fitness landscape* that is easy to navigate, providing a significant advantage to GSGP in terms of evolvability compared to many traditional computational methods.

**Geometric Semantic Crossover (GSC)** is defined as follows. Given two parent functions $T_1, T_2 : \mathbb{R}^n \rightarrow \mathbb{R}$, GSC returns the function:

$$T_{XO} = T_R \cdot T_1 + (1 - T_R) \cdot T_2$$

where $T_R$ is a randomly generated real-valued function whose output lies in the interval $[0, 1]$. This ensures that the offspring's semantic vector is a linear combination of the parents' semantics (Figure 2.15).

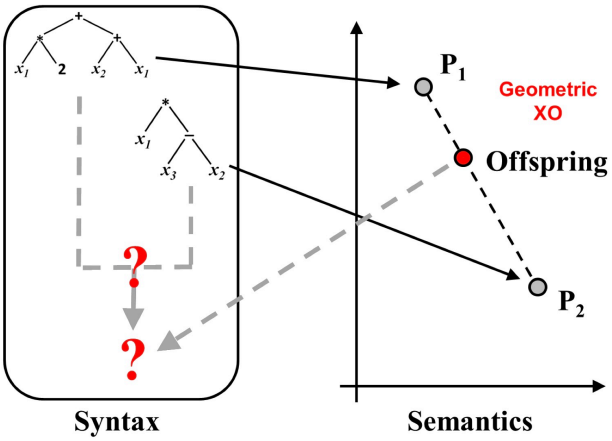


Figure 2.15: Geometric Semantic Crossover: the offspring's semantics lies between the parents in semantic space. *Adapted from* (Vanneschi & Silva, 2023).

As a result, GSC guarantees that the offspring's semantics lies within the region spanned by the parents' semantics and cannot be worse than the worst parent, making the search more stable and directed.

However, despite their theoretical advantages, GSO's present a critical limitation: **exponential growth in individual size**. Each application of GSM or GSC creates an increasingly larger tree, often incorporating entire copies of the parent trees and additional randomly generated subtrees. This leads to severe individual growth in size, rendering standard GSGP impractical for real-world applications.

To address this limitation, several bloat-control strategies have been proposed. Among them, **SLIM_GSGP** introduces a principled approach to preserving the benefits of geometric semantic operators while avoiding the exponential growth of individuals. This method is presented in the following section.

### 2.2.6 Semantic Learning algorithm with Inflate and deflate Mutations (SLIM_GSGP)

To address the extreme individual growth caused by GSGP, (Vanneschi, 2024) proposed a novel framework known as **SLIM_GSGP** (Semantic Learning algorithm with Inflate and deflate Mutations). This approach retains the theoretical strengths of traditional GSGP, particularly its ability to produce a unimodal error surface via the Geometric Semantic Mutation (GSM), while explicitly tackling the issue of fast increase in individual size.

SLIM_GSGP focuses exclusively on mutation-based variation (excluding crossover from initial experiments) and introduces two complementary mutation operators: the traditional GSM (referred to here as the *Inflate Mutation (IGSM)*), and a novel *Deflate Mutation (DGSM)*.

Remembering, GSM is defined as:

$$T_M = T + ms \cdot (T_{R1} - T_{R2})$$

Since $T_{R1}$ and $T_{R2}$ are independently sampled from the same distribution, the subtraction $T_{R1} - T_{R2}$ is **symmetric** and centered around zero, leading to weak perturbations in the offspring's semantics.

Building upon this symmetry , the deflate mutation reuses previously added (inflated) subtrees and subtracts them, effectively shrinking the individual. This operation is not equivalent to backtracking, as it does not reintroduce previously seen individuals. Each deflate mutation perturbs the semantic output in the range $[-ms, ms]$, enabling a controlled reduction in size.

To generalize this concept, SLIM_GSGP explores multiple ways to define the mutation term, particularly functions that map to the interval $[-ms, ms]$. These functions are typically constructed using a sigmoid transformation $S(\cdot)$ applied to random expressions $T_R$, which ensures bounded outputs and improves generalization (Vanneschi et al., 2025). Three function variants are considered:

- **SIG2**: $ms \cdot (S(T_{R1}) - S(T_{R2}))$ - the original GSM formulation using two trees.
- **ABS**: $ms \cdot \left(1 - \frac{2}{1+|T_R|}\right)$ - a single-tree approximation with lower computational cost, though not always zero-centered.
- **SIG1**: $ms \cdot (2 \cdot S(T_R) - 1)$ - a zero-centered variant using a single random tree.

In addition to these semantic perturbation functions, SLIM introduces two ways of applying them to obtain *ball mutations* in semantic space: By adding or subtracting a small value; or by multiplying the existing output by a value close to 1.

Combining the three function variants with the two mutation application strategies results in six distinct SLIM_GSGP variants. These are denoted as **SLIM+** or **SLIM***, depending on whether the operator uses addition or multiplication, followed by the perturbation function used (SIG2, ABS, or SIG1).

A relevant aspect of SLIM_GSGP is that it represents individuals as *linked lists*. This design enables efficient manipulation: inflate mutations are implemented by appending elements to the list, while deflate mutations correspond to deletions. An example of a SLIM+SIG2 genotype, along with its mutated forms (after an inflate followed by a deflate mutation), is illustrated in Figure 2.16.

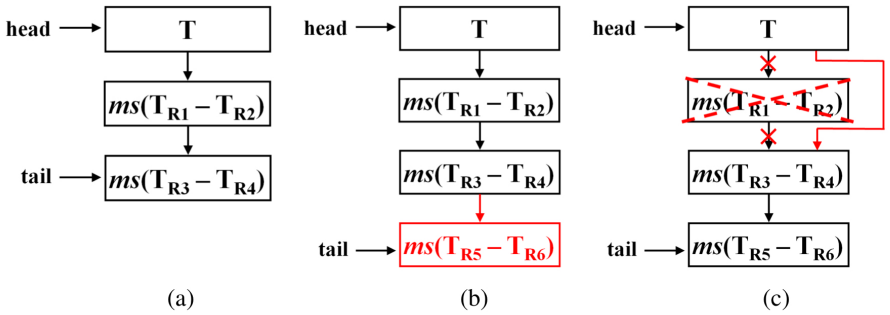


Figure 2.16: An example genotype of a SLIM+SIG2 individual (a), after an inflate mutation (b), and followed by a deflate mutation (c). *Adapted from* (Vanneschi et al., 2025).

This innovation opens the door to generating significantly more compact individuals than both traditional GSGP and standard GP, while preserving semantic awareness and evolvability. The potential and limitations of SLIM_GSGP, as well as related research gaps, are further discussed in the next chapter.

# 3

# Literature Review

In line with the objectives of this thesis, this chapter reviews the existing literature that informs and motivates the research presented herein. The focus is placed on key studies addressing critical challenges and advancements in GSGP, with particular attention to the issues of bloating and the role of the crossover operator.

The literature for this chapter was identified through a systematic search on the SCOPUS database using the following keywords: *bloating AND Genetic Programming*, *GSGP AND crossover*, *geometric AND semantic AND crossover*, and *GSGP AND performance*. Additionally, relevant papers cited in the bibliographies of these works were reviewed and included for further analysis. A total of 68 papers were identified, and this chapter synthesizes the most significant findings from this body of research.

Section 3.1 examines the phenomenon of bloating in GSGP, analyzing both foundational and contemporary works that seek to understand and mitigate this issue. These studies establish a foundation for exploring SLIM_GSGP, offering insights into how this variant may address bloating. Section 3.2 narrows the focus to the crossover operator, specifically its role in GP and its potential when applied to SLIM_GSGP. This section highlights the challenges and opportunities associated with crossover, providing the essential context to situate and justify the contributions of this thesis.

## 3.1 Bloating in GP: Reasoning and Solutions

Koza, in his (1992) publication, demonstrated that Genetic Programming (GP) is a versatile and effective paradigm capable of solving a diverse range of problems across various domains, something considered unlikely for any single computational approach at the time. He introduced the principle that *"structure is a consequence of fitness"* or *"fitness begets structure"*, establishing the foundational ideas that define the field of Genetic Programming today.

Building on this foundation, (Mcphee et al., 2008) examined the standard operators

in GP, particularly the subtree swap crossover, and emphasized the importance of the distribution of programs in the semantic space - the vector of output values of a program over a set of observations - for the success or failure of a search. They argued that future advancements in GP should focus on approaches that move beyond *"blind"* exploration of the syntactic space, advocating instead for methods grounded in theoretical and empirical insights.

This critique of traditional GP's reliance on syntactic search without semantic awareness culminated with the development of Geometric Semantic Genetic Programming (GSGP). (Moraglio et al., 2012) introduced a formal geometric perspective on search operators and program representation, shedding light on the intricate relationship between program syntax and semantics. They formalized the concept of Geometric Semantic Operators (GSOs), which, while operating on the syntactic representation of programs, produce predictable and consistent effects on the semantic properties of the individuals. Notably, Moraglio et al. demonstrated that for problems involving the mapping of input data to known target values, such as regression or classification tasks, GSOs induce fitness landscapes known as cone landscapes, which are inherently easy to navigate. As detailed in (Vanneschi, 2017), the geometric semantic operators generate unimodal error surfaces - fitness landscapes devoid of locally suboptimal solutions. This unique characteristic provides GSGP with a significant advantage in terms of evolvability compared to many traditional computational methods. Moreover, (Vanneschi, Silva, et al., 2014) conducted a deeper analysis of GSO and highlighted an important property: the geometric characteristics of these operators inherently help to limit overfitting.

To understand this conclusion, it is essential to consider the properties of the two key operators. Geometric semantic crossover always generates offspring that lie between the parents in the semantic space, ensuring that the offspring are, at worst, no worse than their parents. Geometric semantic mutation, on the other hand, produces offspring that represent a *"weak"* perturbation of their parent. Importantly, these properties are independent of the data used for evaluation and, as a result, also apply to the test set. While GSOs do not guarantee consistent improvement in test fitness with every application, they do ensure that any potential worsening of test fitness is limited. This characteristic provides a mechanism for controlling overfitting. While limiting overfitting is a significant advantage, it does not negate the fact that, despite its improvements over earlier frameworks, GSGP still has notable limitations.

In the same paper, which evaluates GSGP on pharmacokinetic problems and energy consumption forecasting, the authors revisit a critical observation highlighted by (Moraglio et al., 2012). Specifically, they emphasize that during the evolutionary process, the application of Geometric Semantic Operators (GSOs) results in offspring that inherit the complete structure of the parent along with additional complementary

components, such as random trees and arithmetic operators. This process inevitably causes the offspring to grow exponentially in size, leading to unmanageable population sizes.

As a result, fitness evaluations become increasingly computationally expensive, and the resulting models often grow too complex for human interpretation, making GSOs impractical for many real-world applications unless measures are implemented to control this growth. Over the years, numerous solutions have been explored to address this limitation of GSGP. When introducing this algorithm, Moraglio et al. emphasized the need for code simplification after each generation as a strategy to control the size of individuals while preserving their semantic integrity.

(Vanneschi, Silva, et al., 2014), reflecting on Moraglio et al.'s proposition, concluded that the use of code simplification after each generation would only exacerbate computational expenses. They argued that this approach provides only a partial solution to the problem of program size growth and that its feasibility depends on factors such as the coding language and the primitives involved, potentially making it an exceedingly complex task.

To address this limitation, the authors introduced an efficient implementation of Geometric Semantic Operators (GSOs) that eliminates the need for a simplification step. Originally proposed in (Vanneschi et al., 2013) and further detailed in (Vanneschi, Silva, et al., 2014), this implementation generates the full tree structures of individuals in the initial population, as well as all random trees required for applying operators. For subsequent generations, instead of storing the entire program structures in memory, only the minimal information necessary to reconstruct individuals is maintained. This includes pointers, the names of the operators used, and the semantics of the individuals, which can be efficiently calculated by applying operator definitions to the relevant random trees.

This innovative approach to GSGP significantly enhanced its efficiency in terms of computational time and memory usage, reducing the cost of evolving g generations of n individuals from $O(ng)$ to $O(g)$. This advancement made it feasible, for the first time, to apply GSOs to real-world, complex problems in a practical and scalable manner.

In a similar effort to address the challenge of growing individual sizes in GSGP, (Castelli, Trujillo, & Vanneschi, 2015) extended this efficient GSGP framework by integrating a local search component. Motivated by the inefficiency of GSGP, due to the large number of generations required for convergence and its tendency to overfit, Castelli and colleagues introduced a Local Search Genetic Programming (LSGP) approach. By embedding a local searcher within the Geometric Semantic Mutation operator (GSM-LS), they combined the exploration capability of GSOs with the exploitation strength of the local searcher. This integration improved GSGP's convergence speed toward high-quality solutions, achieving more accurate and reliable models with minimal

impact on execution time. Building on the success of local search applications in GSGP, (Pietropolli et al., 2022) proposed an enhancement that leverages the gradient-based optimizer Adam (Adaptive Moment Estimation). Recognizing that GSOs use randomly selected parameters that remain unoptimized during the evolutionary process, the authors combined GSGP with Adam to capitalize on the complementary strengths of the two methods. While GP drives structural changes in individuals, Adam fine-tunes the parameters of a given structure, something evolutionary processes struggle to achieve efficiently. By using the gradient-based optimizer to refine the parameters used during crossover and mutation (traditionally fixed or random in GSGP), their approach demonstrated improved performance, converging faster to high- quality solutions than traditional GSGP.

Numerous other efforts have been made to address the limitations of GSGP. One such approach (Castelli, Vanneschi, & Popovič, 2015) involves elitist replacement, which incorporates offspring into the population only if they demonstrate better fitness than their parents. This method has been shown to produce models that are *"significantly smaller"* than those generated by traditional GSGP while maintaining comparable performance levels. Nevertheless, the strategies discussed so far represent just a few of the many relevant approaches proposed to overcome the challenges associated with GSGP.

Among the more recent advancements in this matter is SLIM_GSGP. Initially introduced in (Vanneschi, 2024) and further explored in (Vanneschi et al., 2025), the Semantic Learning algorithm with Inflate and Deflate Mutations (SLIM_GSGP) retains the theoretical advantages of traditional GSGP, including its ability to generate an unimodal error surface. It also introduces a novel mutation operator, deflate mutation, designed to produce offspring smaller than their parents. This is achieved by defining geometric semantic mutation (GSM) using subtraction and including a previously added term in the operator. SLIM_GSGP employs genotypes represented as linked lists, allowing inflate mutations to be implemented as simple appends to the list and deflate mutations as deletions.

This implementation results in models that achieve equal or superior performance on unseen data compared to both traditional GSGP and standard GP. Furthermore, these models are substantially smaller than those produced by traditional GSGP and are either smaller or comparable in size to standard GP models, offering the added advantage of compactness and interpretability. This thesis focuses on exploring the potential of this innovative application of GSGP. As a relatively recent implementation, SLIM_GSGP presents a substantial research gap, opening opportunities for extensive investigation. Among these possibilities, this study specifically examines the impact of crossover operators on SLIM_GSGP's performance, aiming to uncover insights into their influence and potential benefits.

## 3.2 Crossover in GP: Impact and Innovations

In the previous section, various studies addressing the limitations of GSGP were discussed, including attempts to improve the Geometric Semantic Crossover (GSC) operator. As highlighted in (Moraglio et al., 2012), GSC has an inherent limitation: it generates offspring by combining the syntax of both parents, causing the size of individuals to grow exponentially over successive generations. Furthermore, as noted in (Castelli et al., 2016), GSC has been shown to perform poorly across a wide range of applications.

In their detailed study, Castelli et al. argue that the poor performance of GSC stems from its geometric properties. Geometric Semantic Crossover consistently produces offspring that remain within the semantic space defined by the parents. As a result, it is limited to generating individuals constrained to the *"boundaries"* of the population's semantic space, known as the convex hull, which significantly restricts its exploratory potential. Through experimental testing, Castelli and coauthors demonstrated that GSGP using only GSC fails to evolve meaningful solutions, with fitness values at the end of a run comparable to those of the initial generation. In contrast, GSGP using both GSC and GSM evolves steadily, improving performance on both training and test sets. Notably, GSGP using only GSM can achieve the same performance as, or in some cases even outperform, GSGP with both GSC and GSM. Furthermore, the study revealed that if the convex hull is far from the target solution, GSC becomes *"virtually useless"* as it cannot produce individuals capable of reaching the global optimum. These findings confirm that GSC provides a negligible contribution to the evolutionary process, with GSM being the primary driver of improvement.This raises an important question:

*What impact can crossovers have on the evolutionary process of SLIM_GSGP?*

Over the years, numerous studies have explored the use of semantics in crossover operators. For a comprehensive analysis of such methods, readers may refer to (Vanneschi, Castelli, & Silva, 2014). Among these studies, several key contributions stand out. In (Krawiec & Lichocki, 2009), Krawiec and Lichocki introduced the concept of semantic distance and perfect fitness-distance correlation to propose a crossover application for GP that approximates Geometric Crossover in the semantic space. Exploiting the property of perfect fitness-distance correlation in the semantic fitness landscape, they developed a method called Approximating Geometric Crossover (AGC). This approach samples a number of offspring generated by a crossover operator and selects the individual with the highest semantic similarity to both parents, termed *"semantically geometric offspring"*, to proceed to the next generation. While this method does not guarantee geometric behavior, it approximates it and emphasizes the benefits of leveraging semantic geometricity.

Building on AGC, (Krawiec, 2012) proposed a class of crossover operators designed to generate semantically intermediate offspring with respect to their parent programs. These operators modify short code fragments (subprograms) to achieve semantic mediality. Two metrics of semantic mediality were introduced: one focused on making offspring semantics geometric relative to the parents, and another aimed at ensuring equidistance from both parents. Both approaches showed performance comparable to non-semantic crossovers, with the latter outperforming all crossovers tested. These contributions laid the foundation for what is now recognized as Geometric Semantic Genetic Programming (GSGP). This progression highlights the transformative role of crossover operators in advancing Genetic Programming, paving the way for further research into their applications in more recent frameworks.

In the case of SLIM_GSGP, Pietropolli et al. (Pietropolli et al., 2025) conducted a study on the impact of crossover operators on this algorithm. Recognizing that the SLIM framework previously relied exclusively on inflate and deflate mutation operators, the authors introduced two novel crossover operators: Swap Crossover (XOSw), which exchanges linked-list components between parents, and Donor Crossover (XODn), which transfers a block from one parent (donor) to another (receiver). These operators aim to enhance the search capabilities of SLIM by incorporating semantic-aware crossovers. Leveraging the linked-list representation of SLIM individuals, these crossovers address the limitations of Geometric Semantic Crossover by allowing offspring to extend beyond the convex hull of the population without unnecessarily increasing size. Experimental results demonstrated that these crossover operators often improve the predictive performance of SLIM, achieving statistically significant fitness gains on unseen data and producing more compact models. The combination of these operators with DGSM proved particularly effective, establishing XOSw and XODn as robust enhancements to traditional GSC and highlighting the advantages of exchanging genetic material during evolution.

Building on these findings, this thesis focuses on further exploring the potential of crossover operators in SLIM_GSGP. Specifically, it aims to conduct a deeper analysis and propose innovations to XODn to maximize the benefits of exchanging genetic material during the evolutionary process.

# 4

# Methodology

This chapter presents the methodology employed in this study, along with the corresponding experimental framework. The primary objective is to evaluate the impact of different crossover methods on SLIM_GSGP, identifying which crossovers yield the best overall performance and whether their impact constitutes a statistically significant improvement over standard SLIM_GSGP without crossover.

Section 4.1 provides a detailed explanation of the tested crossover mechanisms, with Section 4.1.1 focusing on the functioning of XODn and Section 4.1.2 introducing the proposed innovations that led to the development of novel crossover methods tested in this study.

Section 4.2 outlines the experimental setup, with Section 4.2.1 describing the benchmark problems used for evaluation and Section 4.2.2 detailing the complete experimental procedure, including parameter selection and statistical methodologies.

## 4.1 Crossover Implementations in SLIM_GSGP

### 4.1.1 Donor Crossover (XODn): Functioning and Implementation

Given two parent functions:

$$\mathcal{T}_1 = \left(T, \Delta T_1^{(1)}, \ldots, \Delta T_{n1}^{(1)}\right), \quad \mathcal{T}_2 = \left(T, \Delta T_1^{(2)}, \ldots, \Delta T_{n2}^{(2)}\right)$$

Let $\mathcal{T}_1$ be the **donor** and $\mathcal{T}_2$ the **receiver**. The XODn operator produces two offspring, depending on whether the SLIM variant is additive (**A**)or multiplicative (**M**).

$$\mathbf{A}.\mathcal{T}_{\text{XODn}_1} = T_1 + \sum_{i \neq j} \Delta T_i^{(1)}, \quad \mathcal{T}_{\text{XODn}_2} = T_2 + \Delta T_j^{(1)} + \sum_{i=1}^{n_2} \Delta T_i^{(2)}$$

$$\mathbf{M}.\mathcal{T}_{\text{XODn}_1} = T_1 \cdot \prod_{i \neq j} \Delta T_i^{(1)}, \quad \mathcal{T}_{\text{XODn}_2} = T_2 \cdot \Delta T_j^{(1)} \cdot \prod_{i=1}^{n_2} \Delta T_i^{(2)}$$

Where $j$ represents the *donated block index*.

Introduced in (Pietropolli et al., 2025) the Donor Crossover ($XODn$) performs the transfer of a block from a donor individual to a receiver. Figure 4.1 illustrates its general functioning: one parent is randomly designated as the donor and the other as the receiver. A block (e.g.,$\Delta T_3^{(1)}$) is then randomly selected from the donor, removed, and appended to the receiver.

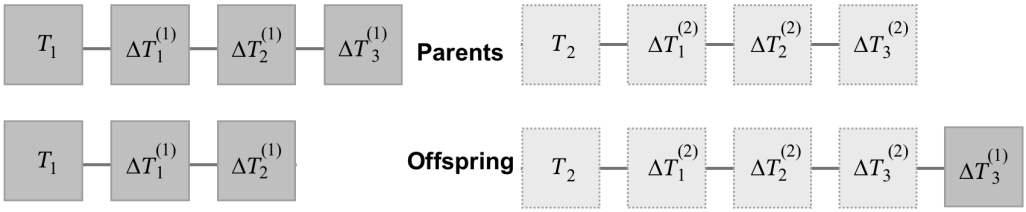


Figure 4.1: Illustrative example of the XODn crossover. The first row shows the two parents, and the second row shows the two offspring generated by XODn.

The pseudo-code for this crossover, shown below, serves as the baseline for all crossover methods developed in this study:

**Algorithm 1:** Donor Crossover (XODn)

**Input:** $individual_1$, $individual_2$
**Output:** $offspring_1$, $offspring_2$
Randomly select one individual as donor, the other as recipient;
**if** *donor has more than one block* **then**
    Choose a random index $j$;
    Remove block $j$ from donor;
    Append block $j$ to recipient;
    **return** $donor, recipient$
**else**
    **return** $individual_1$, $individual_2$
**end**

Conceptually, XODn mimics SLIM mutation operators, acting as DGSM on the donor (which loses a block) and IGSM on the receiver (which gains a block), thereby generating offspring within a radius ms around the parents. Unlike mutation, however, this operator reutilizes existing genetic material from the population rather than introducing a random block, actively promoting genetic exchange during evolution. This characteristic qualifies it as a crossover operator. XODn preserves the geometric properties of GSGP, ensuring a unimodal error surface while overcoming the limitation of Geometric Semantic Crossover (GSC), of restricting offspring to the initial solution space. Furthermore, it maintains offspring sizes close to those of their parents (within ±1 block), effectively preventing uncontrolled growth.

These properties position XODn as a solid foundation for the enhancements introduced in the following section, which aim to refine the crossover mechanism and, consequently, lead to overall performance improvements.

### 4.1.2 Novel Crossovers: Enhancements to XODn

The crossover operators introduced in this study aim to investigate how different design choices influence the behavior of Donor Crossover (XODn) and identify which crossovers contribute to improved performance of the SLIM algorithm. To enable a systematic exploration, the novel crossovers are categorized based on the mechanism used to select the genetic block from the donor individual. This distinction is fundamental, as it determines whether the transferred genetic material is chosen randomly or through an informed, targeted process. This framework results in two main families of crossovers, each embodying a distinct evolutionary strategy:

The first group, **Random Block Crosovers (RBC)**, selects the block to be exchanged purely at random from the donor individual. No information about the block's quality, such as its contribution to fitness or its complexity, is used to guide this selection. Each block in the donor has an equal probability of being chosen. Within this group, some variants also select the donor randomly (e.g., **XODn, DBC, DDC**), while others select the donor based on specific criteria such as fitness or simplicity (e.g., **XOBDn, BDBC, BDDC**). However, what unites them all is the stochastic nature of block selection itself. This randomness enhances the *exploratory* capacity of the algorithm by continually introducing novel and potentially diverse genetic material, even if it is not immediately advantageous. As a result, these crossovers help preserve population diversity and reduce the risk of premature convergence to local optima.

The second group, which we refer to as **Strategic Block Crosovers (SBC)**, incorporates a mechanism for selecting blocks from the donor based on a specific metric or heuristic. Unlike the previous group, the block chosen for transfer is no longer selected randomly but is deliberately picked using available information, such as its fitness impact or semantic properties. Although the donor individual may still be selected randomly or based on a criterion, the key difference lies in the purposeful selection of the donated block. This approach emphasizes *exploitation*, aiming to propagate the most promising genetic material throughout the population. By focusing the search around high-performing genetic structures, these crossovers seek to accelerate convergence toward optimal solutions.

These opreators present a logical drawback: convergence to suboptimal solutions can occur due to a reduction in genetic diversity. By consistently prioritizing and propagating only the currently *"best"* blocks, the algorithm may inadvertently prune potentially valuable genetic material that, while not immediately optimal, could be crucial for escaping local peaks and reaching the global optimum in the long run, a characteristic that Random Block Crossovers are inherently designed to mitigate through their broader exploratory capacity.

Together, Random and Strategic Block Crossovers represent two complementary forces in evolutionary algorithms: **exploration** and **exploitation**. Random Block Crossovers encourage broad, diverse search by introducing unpredictable genetic combinations, while Strategic Block Crossovers refine the search by amplifying high-quality traits. Understanding this dichotomy is essential for analyzing the impact of crossover design on SLIM's search dynamics and overall performance.

#### 4.1.2.1 Random Block Crossovers (RBC)

This category includes crossovers that introduce slight modifications to XODn while maintaining the core mechanism of selecting a random block from the donor and transferring it to a receiver. One such variation is the **Donor Broadcast Crossover (DBC)**.

While XODn follows the process of randomly selecting an individual as the donor and choosing a random block to donate, DBC extends this process by **broadcasting** the chosen block to multiple recipients instead of just one. The purpose of this variation is to evaluate the impact of involving multiple individuals in the genetic material exchange, which could influence the evolutionary process's performance. By donating the *same block* to several recipients, DBC may create offspring more influenced by the donor, potentially preserving beneficial traits across generations. However, this approach also carries the risk of amplifying blocks with negative effects, which could hinder evolutionary progress.

The pseudo-code for this crossover is presented below:

**Algorithm 2:** Donor Broadcast Crossover (DBC)

```
Input: individual_list: list of individuals
Output: Modified individual_list with crossover applied
Randomly select one individual as donor, others as recipients;
if donor has more than one block then
    Select a random index j from the donor;
    Extract block j from donor;
    foreach recipient in recipients do
        Append block j to recipient;
    end
    Remove block j from donor;
    return Modified donor and recipients
else
    return individual_list ;          // No crossover if donor too small
end
```

All further crossover enhancements studied in this work are tested for both the Donor Crossover (XODn) and Donor Broadcast Crossover (DBC) methods. The primary goal of this analysis is to assess the impact of these variations when a single donor donates a block to one recipient (XODn), compared to the scenario where the increased number of individuals present in the genetic material exchange influences the performance and characteristics of the crossover operation (DBC).

As previously discussed, increasing the number of individuals involved in XODn introduces a notable factor: the population becomes more influenced by the donor individual, and the block selected for donation. This raises an important concern:

*What if the genetic material being donated is detrimental? Could propagating "bad" blocks hinder evolution by steering us away from the optimal solution?*

Crossover inherently involves more than one individual, making it crucial to carefully consider which individual possesses the most advantageous traits to pass on to the receivers. The choice of donor has the potential to significantly affect the overall outcomes of the evolutionary process. To address this, one solution is to introduce a crossover variation that incorporates a *donor selection mechanism*. This would allow us to prioritize the propagation of the traits of the most fit individuals from the selected population.

It is important to note that there is no single definitive method for selecting the *"best"* individual, as different selection criteria may lead to different evolutionary outcomes. In this study, two primary metrics for donor selection are explored:

- *Fitness-Based Donor Selection*: The donor is selected based on the fitness value. For minimization, this means selecting the individual with the smallest fitness value, and for maximization problems, selecting the individual with the largest fitness value. This method aims to ensure that the most *"fit"* individual donates its genetic information, potentially propagating superior traits.

- *Size-Based Donor Selection:* The donor is selected as the largest individual. The aim is to reduce the overall size of the population by selecting a large individual to donate a block, helping to control excessive growth within the population.

These donor selection mechanisms are implemented in **Best Donor Crossover (XOBDn)** and **Best Donor Broadcast Crossover (BDBC)**. In XOBDn, the donor is chosen based on the selected metric (fitness or size), and a random block is then removed from it and donated to the recipient. BDBC, an extension of the Donor Broadcast Crossover, also selects the best individual as the donor, but instead of donating to a single recipient, the randomly selected block is donated to all other individuals.

The pseudo-code for XOBDn and BDBC respectively can be found in Algorithm 3 and Algorithm 4.

**Algorithm 3:** Best Donor Crossover (XOBDn)

**Input:** $individual1$, $individual2$,
$measure \in \{\texttt{biggest}, \texttt{min\_fitness}, \texttt{max\_fitness}\}$
**Output:** Two offspring individuals
**if** $measure = \texttt{biggest}$ **then**
  Select **donor** as the largest individual between $individual1$ and $individual2$;
**else if** $measure = \texttt{min\_fitness}$ **then**
  Select **donor** as the individual with minimum fitness;
**end**
**else if** $measure = \texttt{max\_fitness}$ **then**
  Select **donor** as the individual with maximum fitness;
**end**
Assign the other individual as **recipient**;
**if** *donor has more than one block* **then**
  Select a random index $j$ from the donor;
  Extract block $j$ from donor;
  Append block $j$ to recipient;
  **return** *Modified donor and recipient*
**else**
  **return** $individual1$, $individual2$ ; `// No crossover if donor too small`
**end**

While fitness-based selection might seem ideal, it is important to recognize that it does not always guarantee favourable outcomes. Removing a block from the fittest individual may result in the loss of a crucial individual, which could potentially hinder the overall fitness of the population. Additionally, there is no assurance that the selected block, being selected at random, contributes positively to the donor and so, there is no direct association with selecting the best donor and transmitting relevant information to other individuals. The selection of a single block to be donated for multiple individuals leads to a bigger impact of that specific block in the population and overall, could lead to a decrease in diversity in the population.

An approach to mitigate the impact of a single donor block on the population is to *donate different blocks* from the donor to different receivers. This variation increases the probability of selecting relevant blocks to be donated, reducing the influence of any single block on the population. To test this principle, two new variants are introduced: **Distrbuted Donor Crossover (DDC)** and **Distributed Best Donor Crossover (BDDC)**.

**Algorithm 4:** Best Donor Broadcast Crossover (BDBC)

**Input:** $individual\_list$, $measure \in \{\texttt{biggest}, \texttt{min\_fitness}, \texttt{max\_fitness}\}$
**Output:** Modified individual list with donor and recipient offspring
**if** $measure = \texttt{biggest}$ **then**
  Select **donor** as the largest individual in $individual\_list$;
**else if** $measure = \texttt{min\_fitness}$ **then**
  Select **donor** as the individual with minimum fitness;
**end**
**else if** $measure = \texttt{max\_fitness}$ **then**
  Select **donor** as the individual with maximum fitness;
**end**
Assign the remaining individuals as **recipients**;
**if** *donor has more than one block* **then**
  Select a random index $j$ from the donor;
  **foreach** *recipient in recipients* **do**
    Append block $j$ to recipient;
  **end**
  Remove block $j$ from donor;
  **return** *Modified donor and recipients*
**else**
  **return** $individual\_list$ ; `// No crossover if donor too small`
**end**

The **Distributed Donor Crossover** (DDC) is an enhancement of the Donor Broadcast Crossover. Instead of *broadcasting* a single block of genetic material to multiple recipients, DDC **distributes** multiple blocks from the donor to different receivers. Specifically, a donor is selected at random, and for each recipient, a random block from the donor is chosen to be donated. Because multiple blocks are donated in this way, one random block is also removed from the donor to maintain balance in the genetic material exchange. The pseudo-code that illustrates this crossover is shown in Algorithm 5.

**Distributed Best Donor Crossover (BDDC)** follows the same process, but the donor is selected as the *"best"* individual according to specific metrics (for this study, fitness-based and size-based selections). Once the best donor is selected, the same process of choosing a random block to be donated for each receiver and one to be removed from the donor is applied. The pseudo-code for this crossover is presented in Algorithm 6.

While these crossovers reduce the impact of a single donor block on the population, it is important to note that, despite the random selection of blocks, there is still no control over the quality of the genetic information being transmitted to other individuals.

**Algorithm 5:** Distributed Donor Crossover (DDC)

**Input:** $individual\_list$
**Output:** Modified donor and recipients
Select **donor** randomly from $individual\_list$;
Assign the remaining individuals as **recipients**;
**if** *donor has more than one block* **then**
    **foreach** *recipient in recipients* **do**
        Select a random index $j$ the donor;
        Append block $j$ to recipient;
    **end**
    Select a random index $k$ from the donor;
    Remove block $k$ from donor;
    **return** *Modified donor and recipients*
**else**
    **return** $individual\_list$ ; `// No crossover if donor too small`
**end**

**Algorithm 6:** Distributed Best Donor Crossover (BDDC)

**Input:** $individual\_list$, $measure \in$ {`biggest`,`min_fitness`,`max_fitness`}
**Output:** Modified individual list with donor and recipient offspring
**if** $measure =$ `biggest` **then**
    Select **donor** as the largest individual in $individual\_list$;
**else if** $measure =$ `min_fitness` **then**
    Select **donor** as the individual with minimum fitness;
**end**
**else if** $measure =$ `max_fitness` **then**
    Select **donor** as the individual with maximum fitness;
**end**
Assign the remaining individuals as **recipients**;
**if** *donor has more than one block* **then**
    **foreach** *recipient in recipients* **do**
        Select a random index $j$ fromthe donor;
        Append block $j$ to recipient;
    **end**
    Select a random index $k$ from the donor;
    Remove block $k$ from donor;
    **return** *Modified donor and recipients*
**else**
    **return** $individual\_list$ ; `// No crossover if donor too small`
**end**

A potential solution to this issue is to apply selection mechanisms to the donor's blocks, with the aim of propagating the most relevant block and corresponding genetic information.

#### 4.1.2.2 Strategic Block Crossovers (SBC)

To overcome the limitations inherent in random block selection, **Strategic Block Crossovers (SBC)** introduce more deliberate mechanisms to select donor blocks based on their quality or relevance, rather than relying solely on stochastic processes.

A natural way to evaluate a block's relevance is to assess its *impact on the donor individual*. This principle forms the basis of the **Impact-Driven Donor Crossover (IDC)**. In IDC, the donor individual is initially selected at random. From this donor, both the most impactful and least impactful blocks are identified. The impact of a block is quantified by the change in the donor when that block is removed. Three distinct strategies for measuring impact are considered:

- **Fitness Improvement:** the *best* block is the one whose removal causes the *largest deterioration in the donor's fitness*, indicating it contributes most positively to the individual. Conversely, the *worst* block is identified as the one whose removal results in the largest fitness gain (or smallest deterioration), thus having the most detrimental effect.

- **Fitness Disruption:** the *best* block is the one whose removal causes the *greatest absolute change in fitness*, regardless of direction, emphasizing overall disruption rather than only positive or negative effects. The *worst* block causes the smallest absolute change.

- **Semantic Disruption:** The *best* block is the one whose removal induces the *largest change in the donor's semantics*, whereas the *worst* block has minimal semantic impact.

After identifying these blocks, the *best* block is transferred to the recipient individual, promoting the spread of valuable genetic material. Simultaneously, the *worst* block is removed from the donor, facilitating genetic pruning. This dual mechanism balances the exploitation of high-quality building blocks and the elimination of low-quality components. The pseudocode for this operator is presented in Algorithm 7.

Building upon IDC, the **Impact-Driven Best Donor Crossover (BIDC)** refines the approach by also optimizing donor selection. Rather than choosing the donor randomly, BIDC selects the *"best"* individual available as the donor, thereby increasing the likelihood of propagating the most beneficial genetic material within the population (see Algorithm 8).

**Algorithm 7:** Impact-Driven Donor Crossover (IDC)

**Input:** Individuals $A$, $B$; fitness function $F$; target outputs $Y$; impact type $T$
**Output:** Modified individuals $A'$, $B'$

Randomly select one individual as donor $D$, the other as recipient $R$;
**if** *$D$ has more than one block* **then**
  Compute baseline impact T of donor $D$;
  **foreach** *block $b_i$ in $D$* **do**
    Remove $b_i$ from $D$ to form $D_{-i}$;
    Compute impact score $I_i$;
  **end**
  Let $i_{\text{best}} = argmax I_i$;
  Let $i_{\text{worst}} = argmin I_i$;
  Add block $b_{i_{\text{best}}}$ from $D$ to $R$;
  Remove block $b_{i_{\text{worst}}}$ from $D$;
**end**
**return** *updated individuals $A'$, $B'$*

While BIDC maximizes transmission of high-quality information, it carries potential drawbacks. The increased selection pressure may reduce population diversity, leading to premature convergence. Moreover, both IDC and BIDC require continuous evaluation of individual blocks' fitness or semantics, rendering them **computationally more expensive** compared to simpler, structure-based crossover operators.

An alternative perspective on block quality questions the assumption that the *"best"* individual necessarily carries the *"best"* genetic material. Prior approaches select donors based on fixed metrics such as fitness or size, but the complexity of individuals means that the best according to one measure may not yield the best transferable blocks.

This consideration motivates the **Best Block Crossover (BBXO)**. Instead of choosing donors based on a predefined metric or at random, BBXO identifies the *best* block across a population of $N$ individuals. For simplicity, *"best"* here is defined in terms of **Fitness Improvement**: the best block is the one whose removal from the respective individual causes the largest deterioration in fitness, signaling its positive influence. Conversely, the worst block is the one whose removal leads to the greatest fitness improvement (negative impact) from the selected individuals.

Once identified, the individual containing this *best* block is chosen as the donor, and this block is donated to the other individuals (the recipients). To prevent exponential growth in individual size due to repeated block insertions at each generation, each recipient removes its *worst* block, but only if this removal improves its fitness. This strategy controls population growth while maintaining genetic quality and focusing on exploitation. The pseudocode for BBXO is provided in Algorithm 9

**Algorithm 8:** Impact-Driven Best Donor Crossover (BIDC)

**Input:** Individuals $A$, $B$; fitness function $F$; target outputs $Y$; impact type $T$
**Output:** Modified individuals $A'$, $B'$

**if** $measure$ = `biggest` **then**
  Select **donor** $D$ as the largest individual between $A$ and $B$;
**else if** $measure$ = `min_fitness` **then**
  Select **donor** $D$ as the individual with minimum fitness;
**end**
**else if** $measure$ = `max_fitness` **then**
  Select **donor** $D$ as the individual with maximum fitness;
**end**
Assign the other individual as **recipient** $R$;
**if** *$D$ has more than one block* **then**
  Compute baseline impact of donor $D$ using $F$ and $Y$;
  **foreach** *block $b_i$ in $D$* **do**
    Remove $b_i$ from $D$ to form $D_{-i}$;
    Compute impact score $I_i$;
  **end**
  Let $i_{\text{best}} = argmax I_i$;
  Let $i_{\text{worst}} = argmin I_i$;
  Add block $b_{i_{\text{best}}}$ from $D$ to $R$;
  Remove block $b_{i_{\text{worst}}}$ from $D$;
**end**
**return** *updated individuals $A'$, $B'$*

Considering the nature of *Strategic Block Crossovers*, which evaluate and assess individual blocks to determine which should be donated, it follows naturally that these crossover methodologies are computationally more expensive per iteration, for a given individual size, than simpler *Random Block Crossovers*. However, this increased computational cost does not imply inefficiency in the long term. By employing techniques such as removing the worst block from the donor (as in IDC) or from all individuals (as in BBXO), *Strategic Block Crossovers* can produce populations composed of smaller, more streamlined individuals. The reduction in individual size can ultimately compensate for the additional computational effort required per iteration or per individual.

This inherent trade-off between computational complexity and overall performance is a key characteristic of the SBC family and represents an important focus for the experimental phase of this work.

**Algorithm 9:** Best Block Crossover (BBXO)

**Input:** Population $P = \{I_1, I_2, \ldots, I_n\}$; fitness function $F$
**Output:** Modified population $P'$ after crossover

**foreach** *individual* $I \in P$ **do**
  **foreach** *block* $b$ *in* $I$ **do**
    $I_{\setminus b} \leftarrow$ remove block $b$ from $I$;
    $f_{\setminus b} \leftarrow F(I_{\setminus b})$ - calculate fitness of new individual;
    Record impact of the block $(I, b, f_{\setminus b})$;
  **end**
**end**
**foreach** *individual* $I \in P$ **do**
  $b_{\text{best}} \leftarrow$ block whose removal causes largest fitness drop;
  $b_{\text{worst}} \leftarrow$ block whose removal causes fitness improvement;
**end**
Select donor $(I_d, b_d)$ where $b_d$ is the best block across all individuals;
**foreach** *recipient* $I_r \in P \setminus \{I_d\}$ **do**
  Add block $b_d$ to $I_r$;
**end**
**foreach** *individual* $I \in P$ **do**
  Let $b_w$ be worst block of $I$;
  **if** *removing* $b_w$ *improves fitness* **then**
    Remove block $b_w$ from $I$;
  **end**
**end**
**return** *updated population* $P'$;

## 4.2 Experimental Setup

To evaluate the impact of the proposed crossover operators on both fitness and individual size, a well-structured experimental design is essential. This section presents the benchmarks, configurations, and datasets used throughout the study.

### 4.2.1 Evaluation and Problem Benchmarks

Given the objective of this work, to investigate the influence of crossover on SLIM_GSGP, the proposed operators are evaluated within the standard SLIM framework, where crossover is applied alongside both Inflate (IGSM) and Deflate (DGSM) mutation operators. Since the proposed XODn and its variants replicate the behavior of IGSM and DGSM while introducing an explicit genetic material exchange mechanism, we aim to combine the known advantages of mutation with the evolutionary benefits of recombination.

From the six SLIM_GSGP variants introduced by (Vanneschi et al., 2025), we selected two for this study: SLIM+2SIG (additive version) and SLIM*ABS (multiplicative version). These were identified as the best-performing variants of their respective families in (Vanneschi et al., 2025). Because additive and multiplicative models have distinct geometric properties, both are included to allow a broader and more representative assessment of the crossover operators. The performance of each crossover is compared against the mutation-only SLIM (which applies both IGSM and DGSM mutations), allowing us to isolate the effects of introducing novel crossover operators into SLIM_GSGP.

Experiments are conducted on six benchmark symbolic regression problems commonly used in GP research. These datasets were previously analyzed in the context of SLIM_GSGP by (Pietropolli et al., 2025) and (Vanneschi, 2024):

**Toxicity**: 626 features, 274 samples (Archetti et al., 2007); **PPB**: 626 features, 234 samples (Archetti et al., 2007); **Istanbul**: 7 features, 536 samples (Akbilgic et al., 2013); **RBSP (Residential Building Sales Price)**: 107 features, 372 samples (Rafiei, 2015); **Concrete Strength**: 8 features, 1030 samples (Castelli et al., 2013); **Concrete Slump**: 9 features, 102 samples (Yeh, 2009).

### 4.2.2 Experimental Procedure and Statistical Analysis

Following the setup of (Pietropolli et al., 2025), the following parameter configuration is used across all experiments: Trees are initialized via the ramped half-and-half method, with a maximum depth of 6. The function set includes the four standard arithmetic operators $(+, -, *, /)$, with protected division as proposed in (Koza, 1992). The terminal set includes only the input variables (i.e., no constants). Populations consist of 100 individuals evolved over 1000 generations. Selection is performed using tournament selection (size 2), with elitism ensuring that the best individual survives each generation. Fitness is measured using Root Mean Squared Error (RMSE).

For SLIM_GSGP, DGSM is applied with a fixed probability of 0.7, and IGSM with 0.3. All crossover operators are applied with a fixed probability of 20%, with mutation taking place the remaining 80% of the time. These probabilities remain unchanged across experiments to ensure a consistent basis for evaluating crossover behavior. Mutation steps are sampled from a uniform distribution in $[0, 1]$, except for specific datasets where custom ranges are used to follow prior works ((Pietropolli et al., 2025; Vanneschi et al., 2025)): *Toxicity*: $[0, 0.1]$ and *Concrete Strength* and *Slump*: $[0, 3]$. For GSGP, we use the standard configuration of 0.8 crossover and 0.2 mutation.

Each crossover method is tested with the following parameters, as to get a general idea of the impact of hyperparameter selection. For crossovers involving multiple recipients, Random Block Crossovers like DBC and DDC, two settings are tested: 5 and

25 recipients (named `5_XO_Name` and `25_XO_Name`). For best donor selection crossovers (e.g., XOBDn), we test two donor selection strategies: minimum fitness (`min_fit`, given all problems are minimization tasks) and maximum size (`max_size`). For crossovers with both mechanisms like BDBC and BDDC, all combinations of parameters are evaluated.

Strategic Block Crossovers are evaluated as follows: IDC is tested with three impact measures: fitness improvement (`fit_imp`), fitness disruption (`fit_dis`), and semantic disruption (`sem_dis`); BIDC is tested only with fitness improvement (`fit_imp`) and both donor selection methods (`min_fit` and `max_size`); BBXO is tested with 2, 5, 10, and 25 individuals, denoted as `BBXO`, `5_BBXO`, `10_BBXO`, and `25_BBXO`, respectively.

Each configuration is evaluated using Monte Carlo cross-validation with 30 random splits (70% training, 30% testing). All experiments are run 30 times per configuration using the SLIM library (DALabNOVA, 2025). In total, 24 crossover configurations are evaluated. To ensure a thorough assessment, the experimental evaluation is divided into three sequential and interdependent phases. Each phase builds upon the previous one, progressively refining and deepening the analysis to balance performance, interpretability, and efficiency.

**Phase 1: Ranking and Pareto-Based Evaluation** The evaluation begins with a *quantitative* dual-metric analysis of **test fitness** and **individual size** for each crossover and dataset. In **Step 1.1**, the median values from the final generation of each metric, are computed across 30 runs and ranked. The Mann-Whitney U-test (significance level 0.01) is applied to identify crossovers that are statistically indistinguishable from the best performer in each metric, selecting all the following best crossovers until a statistically different crossover is found.

To identify final candidates, we intersect the top sets for both metrics. However, if no crossover is simultaneously top-ranked in both fitness and size, a multi-objective selection is introduced in **Step 1.2**. Here, the non-dominated crossovers (for both metrics) are computed using NSGA-II (Deb et al., 2002), and the best crossover (*Pareto Optima*) is selected by calculating the Euclidean distance to a super-optimal point (minimum observed fitness and size), after Min-Max normalization. Figure 4.2 illustrates this selection process.

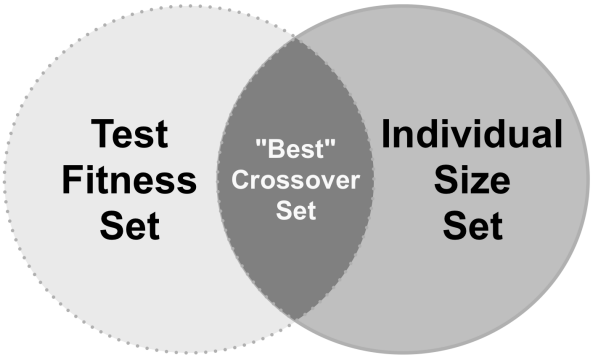


Figure 4.2: Intersection of top-performing crossovers for test fitness and size.

The final selected crossover(s) per dataset follow this decision hierarchy:

1. If a single crossover excels in both metrics (Step 1.1), it is selected;
2. If not, the crossover closest to the Pareto super-optimum (Step 1.2) is chosen;
3. If multiple candidates remain, preference is given to those overlapping both approaches.

**Phase 2: Behavioral Analysis and Final Crossover Selection** While Phase 1 focuses on summary metrics, Phase 2 aims to uncover hidden behavioral patterns across generations that may be masked by aggregation. This phase is essential to ensure that promising yet unstable or bloating crossovers are identified.

In **Step 2.1**, we assess the evolutionary dynamics of each selected crossover using:

- *Fitness Evolution Over Generations*, through line plots;
- *Individual Size at Termination*, via boxplots.

This *qualitative* evaluation helps identify crossovers that converge too early, display overfitting tendencies, or lead to excessive code growth. This is particularly relevant when comparing Random Block Crossovers (RBCs) to Strategic Block Crossovers (SBCs), as the latter often implement anti-bloat mechanisms like worst-block removal.

In **Step 2.2**, we evaluate **computational complexity**, measured as the number of semantic or fitness evaluations required per generation. This step ensures that the selected crossovers are not only performant but also practical for real-world deployment.

Together, these analyses refine a set of top crossovers by incorporating not only their final performance but also their stability and computational feasibility.

**Phase 3: Benchmarking Top Crossovers Against SLIM_GSGP** The final phase benchmarks the final set of crossover-enhanced SLIM_GSGP variants against standard **SLIM_GSGP** without crossover. While Phases 1 and 2 identify the most promising crossovers, Phase 3 addresses the core research question of this thesis:

> *Is crossover a valuable asset to the evolutionary process in SLIM_GSGP? Does the inclusion of genetic material exchange improve performance?*

To answer this, we apply the Mann-Whitney U-test ($\alpha = 0.01$) to compare the final test fitness and size of the best crossover variants against both baselines. This determines whether the improvements are statistically significant or whether crossover can at least match the performance of SLIM_GSGP.

In addition, the same visualization techniques from Phase 2 are used to compare behavioral trends between the selected crossovers and the baselines:

- **Fitness Evolution Over Generations**, to assess performance, convergence speed and overfitting risk;
- **Size at Termination**, to evaluate program size and potential bloat.

This final comparison provides a comprehensive perspective, both *quantitative* and *qualitative*, on the impact of introducing crossover into SLIM_GSGP, highlighting its trade-offs in terms of performance, complexity, and interpretability.

# 5

# Experimental Study

This chapter presents the results (Section 5.1) and discussion (Section 5.2) of the experimental evaluation conducted to assess the impact of novel crossover mechanisms on the performance of SLIM_GSGP. As outlined in Chapter 4, the main objective of this experiment is to identify which crossover methods lead to improvements in model fitness (RMSE) and size, and to determine whether these improvements are statistically significant when compared to SLIM_GSGP without crossover.

The evaluation process follows a three-phase methodology:

- **Phase 1: Ranking and Pareto-Based Evaluation** (Section 5.1.1) - In this phase, 24 crossover configurations are ranked based on their median test RMSE and model size. Statistical testing (Mann-Whitney U-test) identifies the top performers. If no configuration clearly excels on both metrics, a Pareto-based multi-objective analysis (using NSGA-II) is used to select the most balanced crossovers.
- **Phase 2: Behavioral Analysis and Complexity Assessment** (Section 5.1.2) - This phase analyzes the evolutionary behavior of the selected top crossovers by visualizing fitness evolution and size at termination (Step 2.1) and evaluates their computational efficiency using average complexity metrics (Step 2.2). These assessments help refine the final crossover selection by accounting for overfitting, instability, and computational cost.
- **Phase 3: Benchmarking Against standards SLIM_GSGP** (Section 5.1.3) - The final phase compares the top-performing crossover-enhanced variants against the original SLIM_GSGP (without crossover). Statistical tests and visual analysis determine whether the inclusion of crossover offers a significant advantage in terms of performance and generalization.

The following sections detail the outcomes of each evaluation step, providing insights into the relative effectiveness of each crossover method and configuration across the different problem domains.

## 5.1 Experimental Results

### 5.1.1 Phase 1: Ranking and Pareto-Based Evaluation

In an initial phase, the analysis focused on identifying which crossover methods consistently performed better in terms of test fitness and individual size. The main goal is to determine *quantitatively*, a final set of crossovers capable of achieving strong predictive performance while maintaining model compactness, an essential requirement for interpretability and real-world applicability.

In the first step of the evaluation (Step 1.1), for each SLIM variant and dataset, all 24 crossover configurations were ranked from best to worst based on their median test RMSE and individual size at termination, across 30 runs. The top-ranked crossover in each metric served as a reference point, and the Mann–Whitney U test (at a 0.01 significance level) was applied to assess whether other crossovers were statistically indistinguishable from this reference. This procedure resulted in two sets of statistically equivalent top-performing crossovers: one based on fitness and another on size. Full details of the statistical comparisons are provided in Tables I.1 to I.8 in the Annex.

In the second step (Step 1.2), for each SLIM variant and dataset, a Pareto-based multi-objective optimization was performed to identify crossovers that simultaneously balance both evaluation metrics. The non-dominated crossovers were computed using NSGA-II, and a final selection was made by calculating the Euclidean distance to a super-optimum (minimum RMSE and size) and selecting the crossover closest to this point.

The following tables summarize the crossovers selected through the ranking-based procedure (Step 1.1), the Pareto-based optimization (Step 1.2), and the final Best Crossover resulting from the integration of both steps.

| Dataset | Rank Set | Pareto Optima | Best Crossover |
|---|---|---|---|
| **PPB** | 25_BBXO,10_BBXO | XOBDn (max size) | 25_BBXO,10_BBXO |
| **Istanbul** | 25_BBXO | XOBDn (min fit) | 25_BBXO |
| **RBSP** | ∅ | XODn | XODn |
| **Concrete Slump** | ∅ | XODn | XODn |
| **Concrete Strength** | ∅ | 5_BDDC (min fit) | 5_BDDC (min fit) |
| **Toxicity** | 25_BBXO,10_BBXO | 25_BBXO | 25_BBXO |

Table 5.1: Crossovers selected for SLIM*ABS during each selection phase. The *Rank Set* includes crossovers that are the intersection of the fitness and size sets, with ∅ indicating no intersection. *Pareto Optima* lists the crossovers identified as optimal using Pareto fronts. *Best Crossover* refers to the final choice, determined by the selection rules defined previously.

| Dataset | Rank Set | Pareto Optima | Best Crossover |
|---|---|---|---|
| **PPB** | 25_BBXO | XOBDn (max size) | 25_BBXO |
| **Istanbul** | 25_BBXO | 10_BBXO | 25_BBXO |
| **RBSP** | 25_BBXO | 25_BBXO | 25_BBXO |
| **Concrete Slump** | 25_BBXO | 5_BBXO | 25_BBXO |
| **Concrete Strength** | ∅ | BIDC (min fit) | BIDC (min fit) |
| **Toxicity** | 25_BBXO | 10_BBXO | 25_BBXO |

Table 5.2: Crossovers selected for SLIM+SIG2 during each selection phase. The *Rank Set* includes crossovers that are the intersection of the fitness and size sets, with ∅ indicating no intersection. *Pareto Optima* lists the crossovers identified as optimal using Pareto fronts. *Best Crossover* refers to the final choice, determined by the selection rules defined previously.

A closer analysis of the selected crossovers for each SLIM variant (Tables 5.1 and 5.2) reveals several noteworthy patterns. Firstly, regarding the Rank Set, there is a clear predominance of the **Best Block Crossover (BBXO)**, particularly the 25_BBXO variant, which appears in 8 of the final intersection sets, followed by 10_BBXO in 2. Further analysis shows that BBXO consistently dominates the size-based rankings, being exclusively selected as the top performer in terms of individual size across all datasets and SLIM variants. This trend is summarized in Table 5.3, which presents the number of times each crossover was selected across all evaluation sets. While other crossover mechanisms may achieve similar or even superior performance in terms of test fitness, none match BBXO's ability to maintain compact model sizes, especially the 25_BBXO variant. This can be attributed to BBXO's design: it enables genetic exchange by donating the best overall block among the selected individuals to all others, while also integrating an anti-bloating mechanism. This mechanism actively removes blocks that negatively impact fitness, i.e., blocks whose removal leads to performance improvement, helping to control model growth and preserve interpretability.

In cases where BBXO does not dominate both fitness and size metrics, the Pareto Optima are used to select the final best crossover. These optimal sets reveal a significant insight: although BBXO frequently appears in the Rank Set, it is not consistently selected through multi-objective optimization. While it remains the Pareto-optimal choice in 5 out of 12 cases, other crossover mechanisms, such as XOBDn and BIDC, emerge as optimal in the remaining scenarios.

This highlights a fundamental challenge in the analysis: BBXO's dominance in size can overshadow other promising crossovers which, although not the most compact, may still outperform standard SLIM in terms of predictive accuracy and size. Therefore, it is essential to further investigate the individual behaviors of these crossovers.

| Crossover | RMSE | Size | Intersection |
|---|---|---|---|
| XODn | 7 (3) | - | - |
| 5_DBC | 8 (3) | - | - |
| 25_BDC | 7 (3) | - | - |
| XOBDn (min fit) | 8 (4) | - | - |
| 5_BDBC (min fit) | 6 (3) | - | - |
| 25_BDBC (min fit) | 6 (2) | - | - |
| XOBDn (max size) | 7 (3) | - | - |
| 5_BDBC (max size) | 8 (4) | - | - |
| 25_BDBC (max size) | 6 (2) | - | - |
| 5_DDC | 8 (4) | - | - |
| 25_DDC | 5 (1) | - | - |
| 5_BDDC (min fit) | 9 (4) | - | - |
| 25_BDDC (min fit) | 7 (3) | - | - |
| 5_BDDC (max size) | 7 (4) | - | - |
| 25_BDDC (max size) | 4 (0) | - | - |
| IDC (fit imp) | 9 (4) | - | - |
| IDC (fit dis) | 9 (4) | - | - |
| IDC (sem dis) | 8 (4) | - | - |
| BIDC (min fit) | 8 (4) | - | - |
| BIDC (max size) | 8 (4) | - | - |
| BBXO | 8 (4) | - | - |
| 5_BBXO | 9 (4) | - | - |
| 10_BBXO | 8 (3) | 5 (5) | 2 (2) |
| 25_BXO | 8 (3) | 11 (5) | 8 (3) |

Table 5.3: Number of times each crossover was selected as part of the best-performing set. The table shows results for the test fitness (RMSE) and size sets, as well as their intersection. Values indicate the total number of selections for each crossover (across both variants), with SLIM*ABS values shown in parentheses. A dash (–) indicates that the crossover was not selected in the corresponding set.

### 5.1.2 Phase 2: Behavioral Analysis and Complexity Assessment

Building upon the previous dominance analysis, we now shift focus to the behavioral dynamics and computational complexity of selected crossover mechanisms. A key distinction among the studied methods lies in the categorization between **Random Block Crossovers** and **Strategic Block Crossovers**. As previously introduced, Random Block Crossovers, such as XODn, perform minimal modifications by transferring a randomly selected block from donor to receiver. In contrast, Strategic Block Crossovers incorporate heuristics for block selection, often involving semantic or fitness-based evaluations to guide the donation process.

This categorization is important because Strategic Block Crossovers tend to be more computationally complex due to their individual-level evaluation. However, they often incorporate anti-bloat mechanisms (e.g., removing detrimental subtrees) that promote model compactness and improve interpretability. Distinguishing these

groups facilitates a clearer understanding of their trade-offs in terms of performance, generalization, and resource efficiency.

#### Random Block Crossovers (RBC)

Following the methodology outlined in Step 2.1, Figures I.1 and I.2 illustrate the evolution of Train Fitness, Test Fitness, and Elite Node Count for the Random Block Crossovers. Additionally, Figure I.5 shows the distribution of final individual sizes (generation 1000) across 30 runs.

A consistent trend emerges among multi-receiver variants, such as DBC and DDC: while these operators propagate crossover effects across multiple individuals, they tend to induce excessive growth without proportional improvements in fitness. As shown in Figures I.1 and I.2, their increased disruption can promote the survival of unnecessarily large models. Figure I.5 further supports this, indicating increased variability and instability in final model sizes when more individuals receive genetic material per generation.

Interestingly, XOBDn consistently produces the smallest individuals within this group, as corroborated by both visual plots and statistical results (Tables I.5 to I.8). On datasets like *Istanbul* and *PPB*, which are more prone to overfitting, XOBDn achieves better generalization with minimal model growth. Compared to its baseline XODn, it yields comparable or improved fitness and markedly smaller models, demonstrating the benefits of guided donor selection. Additionally, its termination sizes are remarkably stable across runs (Figure I.5), highlighting its robustness.

Based on these insights, XOBDn is selected for continued analysis.

#### Strategic Block Crossovers (SBC)

Applying the same methodology, Figures I.3 and I.4 depict the evolution of fitness and size for the Strategic Block Crossovers. Final model sizes are displayed in Figure I.6.

As expected, these crossovers generally produce smaller models due to their explicit anti-bloat strategies. Despite producing smaller individuals, they often achieve fitness levels comparable to or better than those of Random Block Crossovers. Moreover, they frequently appear in the *Best Rank Sets* for RMSE across diverse datasets and SLIM variants, as shown in Tables I.1–I.8.

The BBXO crossover, in particular, dominates in terms of reducing model size. Its size-reducing effect is amplified with higher participation rates (i.e., more individuals affected per generation), due to its strategy of removing performance-degrading blocks. Conversely, the IDC variants tend to generate larger individuals within this group,

though they exhibit relatively stable test performance. BIDC, a hybrid with donor selection, produces smaller models while maintaining comparable generalization, reinforcing the effectiveness of donor selection.

However, a limitation of BBXO emerges: although highly effective at reducing size, it sometimes suffers from premature convergence, resulting in slightly inferior fitness compared to other crossovers in this group. This underscores a key trade-off between model compactness and predictive performance.

While overfitting is observed in select cases, the general behavior of Strategic Block Crossovers remains robust and consistent with earlier observations.

### Complexity Analysis

Having narrowed down the most promising candidates, namely all Strategic Block Crossovers and XOBDn, we now examine their computational complexity. Figure 5.1 presents the total number of fitness and semantic evaluations over 1000 generations as a proxy for algorithmic effort, for SLIM+SIG2.

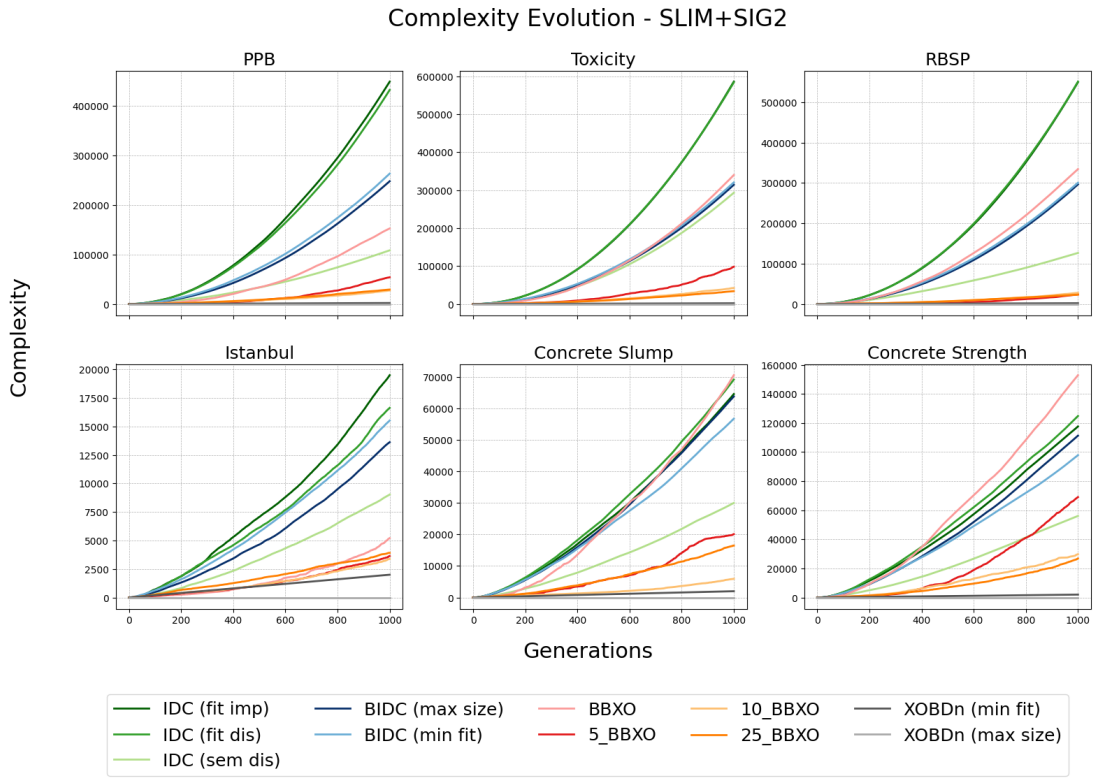


Figure 5.1: Complexity Study of **Strategic Block Crossovers** and **XOBDn**. Lines show the evolution of the training fitness, testing fitness, and elite node count for SLIM+SIG2, across generations, with median values across 30 runs for each crossover operator.

Results are largely in line with expectations (for SLIM*ABS results see Figure I.7). Crossovers that generate larger individuals tend to exhibit greater computational costs. XOBDn, as a member of the Random Block group, maintains significantly lower complexity compared to Strategic counterparts. Among the Strategic group, BBXO is the most efficient, with complexity levels most comparable to XOBDn.

Notably, IDC (sem dis), which relies solely on semantic distance rather than fitness evaluations, is the least complex among the IDC crossovers. Its simplicity does not appear to compromise performance, suggesting that semantic guidance alone can be a valuable strategy for efficient and effective evolution.

All Strategic Block Crossovers, along with XOBDn and XODn (used as a baseline reference), will be included in the next phase of the analysis, where they are compared against standard SLIM_GSGP.

### 5.1.3 Phase 3: Benchmarking Against standards SLIM_GSGP

Given the overall behavior of the previously analyzed crossover operators, it is now crucial to assess their performance relative to the *SLIM_GSGP* baseline, which uses only mutation operators. Each selected crossover configuration was statistically compared to *SLIM_GSGP* for both SLIM variants and across all studied datasets. These comparisons were conducted using the Mann-Whitney U test (with $\alpha = 0.01$) to evaluate whether the performance differences in test fitness and model size were statistically significant. The results of this benchmarking analysis are summarized in Table 5.4, while Tables I.9-I.16 in the Annex provide detailed statistical information.

| **SLIM VERSION** | **SLIM*ABS** | **SLIM+SIG2** | **Total** | **SLIM*ABS** | **SLIM+SIG2** | **Total** |
|---|---|---|---|---|---|---|
| **Crossover** | **vs SLIM RMSE** | | | **vs SLIM Size** | | |
| XODn | 5 (3) | 6 (1) | 11 (4) | 0 (0) | 0 (0) | 0 (0) |
| XOBDn (min fit) | 6 (2) | 6 (0) | 12 (2) | 6 (0) | 6 (4) | 12 (4) |
| XOBDn (max size) | 5 (1) | 6 (1) | 11 (2) | 6 (4) | 6 (3) | 12 (7) |
| IDC (fit imp) | 5 (1) | 6 (2) | 11 (3) | 4 (0) | 3 (2) | 7 (2) |
| IDC (fit dis) | 6 (0) | 6 (1) | 12 (1) | 3 (0) | 3 (2) | 6 (2) |
| IDC (sem dis) | 6 (0) | 6 (0) | 12 (0) | 5 (2) | 5 (4) | 10 (6) |
| BIDC (min fit) | 5 (0) | 6 (2) | 11 (2) | 6 (2) | 3 (3) | 9 (5) |
| BIDC (max size) | 5 (1) | 6 (1) | 11 (2) | 6 (2) | 3 (3) | 9 (5) |
| BBXO | 5 (0) | 6 (0) | 11 (0) | 6 (3) | 5 (2) | 11 (5) |
| 5_BBXO | 4 (0) | 5 (1) | 9 (1) | 6 (5) | 6 (5) | 12 (10) |
| 10_BBXO | 4 (0) | 5 (1) | 9 (1) | 6 (6) | 6 (6) | 12 (12) |
| 25_BBXO | 3 (1) | 5 (1) | 8 (2) | 6 (6) | 6 (6) | 12 (12) |

Table 5.4: Number of cases in wich each Crossover returned statistically significant better or equal solutions (strictly better between parenthesis) than SLIM in terms of test RMSE and Size.

Observing Table 5.4, several interesting patterns emerge. Although **BBXO** was consistently highlighted in the crossover selection process discussed in Section 5.1.1, it now becomes evident that its strong performance in reducing model size may come at the cost of overall test fitness. Among the tested crossovers, the *BBXO* class exhibits the lowest number of datasets where it is statistically equal or better than *SLIM* in test RMSE, only 8 to 11 out of 12 cases. Still, its impact on model size is remarkable: all BBXO variants outperform or match SLIM across all 12 datasets (BBXO in 11 out of 12), with the 10_BBXO and 25_BBXO variants showing the most consistent superiority in reducing individual size. Although their fitness results are less impressive, achieving better fitness than SLIM in just 1 or 2 datasets, these versions prove that the expected fitness degradation for smaller individuals is not guaranteed and may depend on problem characteristics. This inconsistency across datasets suggests that BBXO's effectiveness is highly context-dependent and warrants deeper investigation.

This behavior aligns with the trends observed in Figure I.4, where BBXO variants showed suboptimal performance in datasets like *PPB* and *Concrete Strength*, but achieved notable success in others such as *RBSP* and *Toxicity*. This bipolar behavior highlights BBXO's dual nature: while it can hinder performance in some tasks due to overly aggressive size reduction, it also excels in cases where SLIM struggles to find optimal solutions. These findings may indicate that exploitation-heavy strategies like BBXO offer benefits in particularly challenging or noisy search spaces.

As for **XODn**, the results are consistent with prior analyses: while it performs competitively in test fitness, it demonstrates poor performance in individual size, highlighting a strong tendency toward bloating. Without additional anti-bloat mechanisms, *XODn* appears unsuitable for scenarios requiring compact models.

**XOBDn**, on the other hand, emerges as one of the most impactful crossovers in this study. Both variants consistently achieve equal or smaller individual sizes compared to SLIM across all datasets. The version selecting the fittest donor improves size in 4 datasets; the version using the largest donor improves in 7. While these gains are smaller than those of BBXO, they are achieved without any built-in size control, suggesting a promising balance. In terms of test fitness, the minimum fitness version of XOBDn performs equal or better in all 12 datasets (better in 2), while the size-focused variant performs equal or better in 11 (better in 2). These results support the idea that selection mechanisms, such as donor criteria, can strongly influence crossover behavior and should be explored further.

Taken together, the **BBXO** and **XOBDn** classes stand out as the most impactful crossovers tested. BBXO offers unmatched size reduction, whereas XOBDn provides consistent, stable improvements over SLIM with fewer trade-offs. These results suggest

that XOBDn is a promising candidate for integrating crossover in SLIM-based models, offering improvements with minimal risk of performance loss.

To further explore this observation, a final graphical comparison is performed between SLIM, BBXO, and XOBDn. Figures I.8 and I.9 illustrate the evolution of train fitness, test fitness, and elite node count. Figure I.10 complements this with the distribution of final individual sizes at generation 1000 across 30 runs.

A simple analysis of these plots confirms the findings discussed in this chapter: across most datasets, there is consistently at least one crossover (BBXO or XOBDn) that matches or outperforms SLIM on one or more key dimensions.

These results confirm that integrating well-designed crossover operators into SLIM_GSGP can provide tangible improvements in model compactness and generalization, marking a significant step toward more effective and interpretable symbolic models.

## 5.2 Discussion of Results

This work presents a comprehensive study on different crossover methods and strategies for SLIM_GSGP. A wide range of mechanisms were tested, making it essential to highlight the most impactful findings. A key outcome was the identification of consistent performance patterns across datasets and SLIM variants. Several insights emerged from the comparative analysis.

Notably, **donor selection mechanisms** (e.g., XOBDn and BIDC) facilitated the propagation of high-quality genetic material while implicitly controlling bloating. These mechanisms led to more compact models that matched or outperformed their simpler counterparts and exhibited more stable performance across multiple runs.

In contrast, **multi-receiver mechanisms** (e.g., DBC and DDC) tended to generate larger individuals without consistent gains in fitness. This behavior is likely due to the increased growth caused by the simultaneous distribution of similar genetic material to multiple offspring in each generation. Moreover, these operators showed greater variability in performance across runs, suggesting a trade-off between exploration and stability.

Among all random block operators (RBC), **XOBDn** stood out as particularly effective. It achieved fitness levels comparable to other methods while producing significantly smaller models than typical random block selectors. Its success appears closely tied to the inclusion of donor selection. Meanwhile, **BBXO** frequently yielded the smallest

individuals (in most cases considerably smaller) but displayed higher variance in fitness outcomes. This trade-off suggests that while aggressive model simplification is possible, it may come at the cost of generalization. These findings emphasize the potential of *crossover parameter tuning*, which could help adjust the simplification-generalization balance for better real-world applicability.

When it comes to crossover complexity, **BBXO** also raises a critical point: although computational efficiency is important, more complex crossover mechanisms can lead to overall better results. BBXO, while algorithmically more involved, has a complexity not far from XOBDn, especially when compared to other strategic block crossovers. Its powerful anti-bloating strategies produce individuals significantly smaller than any other operator, which minimizes the practical cost of its complexity. This highlights the importance of integrating explicit anti-bloating mechanisms into crossover design.

An honorable mention goes to **IDC (sem dis)**, the only semantic-aware crossover in the *Strategic Block Crossovers* group. Unlike its fitness-focused counterparts, IDC selects blocks based on semantic disruption only, and still achieved performance on par or even superior. This suggests that semantic-aware strategies may offer meaningful enhancements, opening a promising avenue for future research.

The dominance of both a **Random Block Crossover** (XOBDn) and a **Strategic Block Crossover** (BBXO) illustrates the importance of balancing exploration and exploitation in the evolutionary process. Their complementary strengths show that no single strategy is universally superior and that hybrid or adaptive approaches may yield further improvements.

Overall, when compared to the baseline *SLIM_GSGP*, which relies solely on mutation, the proposed crossover operators frequently achieved better test RMSE and smaller (in some cases significantly smaller) models. These results reinforce the role of genetic material exchange in guiding SLIM's evolution and demonstrate that crossovers can offer improved generalization while promoting more interpretable, deployment-ready models.

From a research perspective, this study reinforces the pivotal role of crossover design in shaping both search dynamics and model complexity in GSGP. It offers a blueprint for developing crossover operators that are not only effective but also efficient. From a practical perspective, where model size, stability, and computational cost are crucial, crossovers like *XOBDn* offer promising solutions for evolving compact, generalizable models. On the other hand, *BBXO* demonstrates how powerful, impact-driven crossovers can enhance SLIM_GSGP's potential for real-world applications by producing models significantly smaller than those evolved through mutation alone.

This work benefited from a rigorous experimental design. Crossovers were benchmarked systematically across multiple datasets and SLIM variants, using consistent protocols. The use of non-parametric statistical tests and detailed visual analyses (e.g., fitness and size evolution plots) enabled the identification of meaningful behavioral patterns and differences between operators.

Despite these contributions, several limitations must be acknowledged:

- **Parameter Tuning:** This study focused on a broad comparison of crossover mechanisms, with only a light exploration of parameter ranges. A more focused analysis is needed to fully understand the potential of these operators when optimized for specific tasks.
- **Dataset Scope:** Although the six regression datasets were diverse, they may not fully represent the variety of challenges encountered in real-world symbolic regression applications.
- **Limited Internal Analysis:** The study primarily analyzed external behaviors (e.g., fitness, model size), with less emphasis on internal dynamics such as redundancy and functional complexity.

These limitations point to several promising directions for future work:

- **Crossover Parameter Tuning:** Deeper exploration of hyperparameters (e.g., crossover rate, number of receivers, selection metric) may lead to better performance and a more adaptive trade-off between speed, complexity, and generalization.
- **Functional Complexity Control:** New strategies could go beyond size reduction by targeting functional redundancy and behavioral diversity, promoting more meaningful simplification.
- **Anti-Bloating Strategies:** Further development and systematic incorporation of anti-bloating mechanisms could improve efficiency and generalization across different crossover types.
- **Semantic-Aware Crossovers:** Building on the insights from IDC, future operators should explore semantic awareness as a primary design principle to enhance performance and interpretability.

In summary, this study provides strong evidence against the continued avoidance of crossover in GSGP-based approaches. Operators like *XOBDn* show that well-designed crossovers can evolve smaller, high-performing models, while *BBXO* reveals the transformative potential of integrating anti-bloating and fitness-aware strategies

into crossover design. Together, these findings contribute to a more effective, compact, and interpretable approach to symbolic regression, bringing SLIM_GSGP one step closer to real-world deployment in complex, resource-constrained environments.

# 6

# Conclusions

This thesis addressed a long-standing limitation in Geometric Semantic Genetic Programming (GSGP): the exclusion of crossover in SLIM_GSGP due to its historically poor performance and its contribution to bloating. While SLIM_GSGP had already proven effective through semantic-aware mutation mechanisms, crossover, one of the core evolutionary operators, remained underexplored. This work revisited crossover not as an afterthought but as a potentially valuable and viable asset to SLIM_GSGP evolution, capable of reinforcing the search process through controlled genetic material exchange.

To do this, several crossover mechanisms were designed and implemented, each aligned with the internal linked-list representation of SLIM. These included random block exchanges, donor-based strategies, and block selectors, each crafted to promote genetic exchange while looking to control structural growth - driving both exploration and exploitation. Their performance was evaluated across two SLIM variants (SLIM+2SIG and SLIM*ABS) and six benchmark symbolic regression problems. The evaluation focused on predictive performance, model size, and evolutionary behavior.

Results clearly demonstrated that crossover can be effectively reintroduced into SLIM_GSGP with measurable benefits. Operators such as Best Donor Crossover (XOBDn), which leverages donor selection, consistently produced compact models without sacrificing generalization. Likewise, Best Block Crossover (BBXO), designed with an anti-bloat mechanism, excelled in minimizing model size and improving interpretability. Against standard SLIM_GSGP (without crossover), many of the proposed crossover-enhanced variants matched or exceeded their counterparts in predictive performance while delivering smaller and more interpretable models.

These findings challenge the assumption that crossover is incompatible with semantic GP, showing instead that, when semantically aligned, it can enhance performance, accelerate convergence, and complement SLIM's mutation-driven evolution.

Despite these contributions, some limitations remain. First, the study focused on general operator behavior and only lightly explored parameter tuning. While some exploratory tuning was conducted, a systematic and comprehensive analysis of crossover hyperparameters (e.g., receiver count, donor selection strategy) was not performed, which may limit the observed performance of certain operators. Lastly, the analysis centered on external performance indicators such as test RMSE and model size, with limited examination of internal dynamics like semantic diversity, block redundancy, or functional complexity.

Building on the findings of this study, several avenues for future research emerge. First, deeper exploration of crossover parameter tuning could uncover more adaptive trade-offs between model complexity, generalization, and computational efficiency. Fine-tuning these parameters may further optimize performance, especially in domain-specific contexts. Second, the development and systematic integration of anti-bloating mechanisms should be prioritized. As demonstrated by *BBXO*, such mechanisms can drastically reduce model size without sacrificing performance, enhancing the practicality of crossover-driven SLIM variants. Third, beyond controlling individual size, future strategies could incorporate functional complexity control, targeting block-level redundancy and encouraging behavioral diversity to yield more meaningfully simplified models. Finally, this work highlights the untapped potential of semantic-aware crossovers. Inspired by the promising results of the *IDC* operator, future efforts should analyse semantic-space exploration designs. This shift in perspective could enable the development of crossover operators that are not only more interpretable and generalizable but also better aligned with the foundational goals of Geometric Semantic GP.

This thesis demonstrates that crossover is not only compatible with SLIM_GSGP but can serve as a powerful driver of its evolutionary process. Through semantically aligned design and structural awareness, crossover operators can enrich the search space, improve model compactness, and enhance generalization, advancing both the theory and practical applicability of SLIM_GSGP. These findings challenge prior assumptions and open the door to further innovation in semantics-aware evolutionary computation.

Ultimately, this work redefines the role of crossover in Geometric Semantic Genetic Programming, not as a legacy operator to be avoided, but as a carefully designed, semantically aligned mechanism that can drive the next generation of interpretable, compact, and high-performing symbolic models.

# Bibliography


Akbilgic, O., Bozdogan, H., Balaban, E., & Balaban, M. (2013). A novel hybrid rbf neural networks model as a forecaster. *Statistics and Computing*, *24*. https://doi.org/10.1007/s11222-013-9375-7 (cit. on p. 41).

Archetti, F., Lanzeni, S., Messina, E., & Vanneschi, L. (2007). Genetic programming for computational pharmacokinetics in drug discovery and development [Place: USA. Publisher: Kluwer Academic Publishers]. *Genetic Programming and Evolvable Machines*, *8*(4), 413–432. https://doi.org/10.1007/s10710-007-9040-z (cit. on p. 41).

Castelli, M., Manzoni, L., Gonçalves, I., Vanneschi, L., Trujillo, L., & Silva, S. (2016). An Analysis of Geometric Semantic Crossover: A Computational Geometry Approach, 201–208. https://doi.org/10.5220/0006056402010208 (cit. on pp. 2, 27).

Castelli, M., Trujillo, L., & Vanneschi, L. (2015). Energy Consumption Forecasting Using Semantic-Based Genetic Programming with Local Search Optimizer. *Computational Intelligence and Neuroscience*, *2015*, 1–8. https://doi.org/10.1155/2015/971908 (cit. on p. 25).

Castelli, M., Vanneschi, L., & Popovič, A. (2015). Controlling Individuals Growth in Semantic Genetic Programming through Elitist Replacement. *Computational Intelligence and Neuroscience*, *2016*, 1–12. https://doi.org/10.1155/2016/8326760 (cit. on p. 26).

Castelli, M., Vanneschi, L., & Silva, S. (2013). Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. *Expert Systems with Applications*, *40*(17), 6856–6862. https://doi.org/10.1016/j.eswa.2013.06.037 (cit. on p. 41).

DALabNOVA. (2025, May 7). *DALabNOVA/slim*. https://github.com/DALabNOVA/slim (cit. on p. 42).

Darwin, C. (1859). *On the origin of species by means of natural selection*. John Murray. (Cit. on pp. 7, 10).

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017 (cit. on p. 42).

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1964). On the evolution of artificial intelligence. *Proceedings of the Fifth National Symposium on Human Factors in Electronics*, 63–76 (cit. on p. 8).

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press. (Cit. on pp. 8, 10).

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. MIT Press. (Cit. on pp. 1, 8, 15, 41).

Krawiec, K. (2012). Medial Crossovers for Genetic Programming, 61–72. https://doi.org/10.1007/978-3-642-29139-5_6 (cit. on p. 28).

Krawiec, K., & Lichocki, P. (2009). Approximating geometric crossover in semantic space [event-place: Montreal, Québec, Canada]. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, 987–994. https://doi.org/10.1145/1569901.1570036 (cit. on p. 27).

Lourenço, J. M. (2021). *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. iii).

Mcphee, N., Ohs, B., & Hutchison, T. (2008). Semantic Building Blocks in Genetic Programming, 134–145. https://doi.org/10.1007/978-3-540-78671-9_12 (cit. on p. 23).

Mitchell, T. M. (1997). *Machine learning* (Vol. 1). McGraw-hill New York. (Cit. on p. 5).

Moraglio, A., Krawiec, K., & Johnson, C. (2012). Geometric Semantic Genetic Programming. *7491*, 21–31. https://doi.org/10.1007/978-3-642-32937-1_3 (cit. on pp. 1, 13, 17, 24, 27).

Pietropolli, G., Farinati, D., Manzoni, L., Castelli, M., Silva, S., & Vanneschi, L. (2025). Introducing crossover in SLIM-GSGP. *Genetic Programming*, 103–119 (cit. on pp. 28, 30, 41).

Pietropolli, G., Manzoni, L., Paoletti, A., & Castelli, M. (2022, April). Combining Geometric Semantic GP with Gradient-Descent Optimization. https://doi.org/10.1007/978-3-031-02056-8_2 (cit. on p. 26).

Rafiei, M. (2015). *Residential building* [[Dataset]]. UCI Machine Learning Repository. https://doi.org/10.24432/C5S896 (cit. on p. 41).

Turning, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433 (cit. on p. 5).

Vanneschi, L. (2017). An Introduction to Geometric Semantic Genetic Programming. In O. Schütze, L. Trujillo, P. Legrand, & Y. Maldonado (Eds.), *NEO 2015: Results of the Numerical and Evolutionary Optimization Workshop NEO 2015 held at September 23-25 2015 in Tijuana, Mexico* (pp. 3–42). Springer International Publishing. https://doi.org/10.1007/978-3-319-44003-3_1 (cit. on p. 24).

Vanneschi, L. (2024). SLIM_gsgp: The Non-bloating Geometric Semantic Genetic Programming. In M. Giacobini, B. Xue, & L. Manzoni (Eds.), *Genetic Programming* (pp. 125–141). Springer Nature Switzerland. (Cit. on pp. 1, 20, 26, 41).

Vanneschi, L., Castelli, M., Manzoni, L., & Silva, S. (2013). A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics. In K. Krawiec, A. Moraglio, T. Hu, A. Ş. Etaner-Uyar, & B. Hu (Eds.), *Genetic Programming* (pp. 205–216). Springer Berlin Heidelberg. (Cit. on p. 25).

Vanneschi, L., Castelli, M., & Silva, S. (2014). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, *15*. https://doi.org/10.1007/s10710-013-9210-0 (cit. on p. 27).

Vanneschi, L., Farinati, D., Rasteiro, D., Rosenfeld, L., Pietropolli, G., & Silva, S. (2025). Exploring Non-bloating Geometric Semantic Genetic Programming. *SpringerLink*, 237–258. https://doi.org/10.1007/978-981-96-0077-9_12 (cit. on pp. 2, 20, 21, 26, 41).

Vanneschi, L., & Silva, S. (2023, January). *Lectures on intelligent systems*. Springer, Cham. https://doi.org/https://doi.org/10.1007/978-3-031-17922-8 (cit. on pp. 6, 8–10, 18, 19).

Vanneschi, L., Silva, S., Castelli, M., & Manzoni, L. (2014). Geometric Semantic Genetic Programming for Real Life Applications [ISBN: 978-1-4939-0374-0]. *Genetic Programming Theory and Practice XI*, 191–209. https://doi.org/10.1007/978-1-4939-0375-7_11 (cit. on pp. 24, 25).

Yeh, I.-C. (2009). Simulation of concrete slump using neural networks. *Construction Materials*, *162*, 11–18 (cit. on p. 41).

# I

# Annex 1

| Datasets | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossovers** | **RMSE** | **P-value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 29,275 | 0,075 | 0,0161 | 0,054 | **49,263** | **BEST** |
| 5_DBC | 30,747 | 0,005 | 0,0163 | 0,012 | 57,750 | 0,246 |
| 25_DBC | 30,386 | 0,217 | 0,0159 | 0,061 | 65,845 | 0,005 |
| XOBDn (min fit) | 28,012 | 0,455 | **0,0152** | **BEST** | 53,428 | 0,290 |
| 5_BDBC (min fit) | 29,584 | 0,112 | 0,0165 | 0,022 | 57,384 | 0,158 |
| 25_BDBC (min fit) | 29,422 | 0,105 | 0,0162 | 0,033 | 66,764 | 0,011 |
| XOBDn (max size) | **27,468** | **BEST** | 0,0156 | 0,540 | 54,390 | 0,171 |
| 5_BDBC (max size) | 29,406 | 0,075 | 0,0168 | 0,001 | 58,237 | 0,119 |
| 25_BDBC (max size) | 32,635 | 0,002 | 0,0157 | 0,290 | 72,153 | 0,000 |
| 5_DDC | 29,721 | 0,025 | 0,0166 | 0,012 | 52,798 | 0,728 |
| 25_DDC | 30,757 | 0,012 | 0,0164 | 0,008 | 72,721 | 0,001 |
| 5_BDDC (min fit) | 30,601 | 0,068 | 0,0161 | 0,016 | 51,766 | 0,819 |
| 25_BDDC (min fit) | 29,300 | 0,258 | 0,0158 | 0,186 | 62,959 | 0,007 |
| 5_BDDC (max size) | 29,413 | 0,126 | 0,0168 | 0,001 | 53,975 | 0,610 |
| 25_BDDC (max size) | 31,052 | 0,047 | 0,0160 | 0,003 | 67,937 | 0,001 |
| IDC (fit imp) | 27,632 | 0,739 | 0,0156 | 0,217 | 54,129 | 0,520 |
| IDC (fit dis) | 30,399 | 0,115 | 0,0157 | 0,158 | 64,878 | 0,006 |
| IDC (sem dis) | 27,657 | 0,773 | 0,0155 | 0,333 | 65,615 | 0,007 |
| BIDC (min fit) | 27,626 | 0,877 | 0,0156 | 0,420 | 66,374 | 0,013 |
| BIDC (max size) | 29,614 | 0,620 | 0,0154 | 0,695 | 61,963 | 0,030 |
| BBXO | 28,014 | 0,728 | 0,0153 | 0,569 | 73,511 | 0,000 |
| 5_BBXO | 30,017 | 0,041 | 0,0157 | 0,464 | 75,925 | 0,000 |
| 10_BBXO | 28,707 | 0,333 | 0,0157 | 0,663 | 81,424 | 0,000 |
| 25_BXO | 29,522 | 0,333 | 0,0155 | 0,304 | 88,514 | 0,000 |

Table I.1: Test Root Mean Square Error (RMSE) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossovers** | **RMSE** | **P-Value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 7,903 | 0,807 | 8,853 | 3,37E-04 | 2568,961 | 1,17E-03 |
| 5_DBC | **7,753** | **BEST** | 8,065 | 0,158 | 2743,041 | 2,49E-06 |
| 25_DBC | 8,584 | 0,066 | 9,559 | 5,00E-09 | 2653,495 | 5,87E-04 |
| XOBDn (min fit) | 8,944 | 0,008 | 8,973 | 2,68E-06 | 2347,260 | 0,145 |
| 5_BDBC (min fit) | 7,974 | 0,333 | 8,388 | 4,43E-03 | 2533,821 | 2,01E-04 |
| 25_BDBC (min fit) | 8,518 | 0,082 | 9,557 | 4,18E-09 | 2588,605 | 3,37E-05 |
| XOBDn (max size) | 8,889 | 0,009 | 9,499 | 1,70E-08 | 2523,960 | 4,64E-03 |
| 5_BDBC (max size) | 7,926 | 0,728 | 8,294 | 0,018 | 2812,322 | 5,09E-06 |
| 25_BDBC (max size) | 8,531 | 0,080 | 11,045 | 2,87E-10 | 2491,830 | 3,56E-04 |
| 5_DDC | 8,067 | 0,206 | 8,274 | 0,102 | 2834,154 | 4,42E-06 |
| 25_DDC | 8,109 | 0,158 | 9,155 | 9,53E-07 | 2717,851 | 1,25E-05 |
| 5_BDDC (min fit) | 7,812 | 0,559 | 8,170 | 0,137 | 2670,111 | 4,35E-05 |
| 25_BDDC (min fit) | 8,317 | 0,075 | 9,083 | 1,17E-05 | 2753,205 | 1,49E-06 |
| 5_BDDC (max size) | 7,980 | 0,340 | **7,899** | **BEST** | 2871,991 | 1,01E-08 |
| 25_BDDC (max size) | 9,084 | 0,004 | 17,028 | 2,15E-10 | 2792,901 | 2,68E-06 |
| IDC (fit imp) | 8,333 | 0,171 | 9,342 | 3,01E-07 | 2583,744 | 5,87E-04 |
| IDC (fit dis) | 8,285 | 0,191 | 9,849 | 6,52E-09 | 2328,190 | 0,023 |
| IDC (sem dis) | 8,336 | 0,258 | 9,343 | 9,06E-08 | 2342,122 | 0,112 |
| BIDC (min fit) | 8,407 | 0,034 | 9,641 | 1,41E-09 | 2323,849 | 0,047 |
| BIDC (max size) | 8,557 | 0,064 | 9,440 | 1,85E-08 | 2635,632 | 2,27E-03 |
| BBXO | 8,255 | 0,068 | 11,426 | 6,70E-11 | 2319,648 | 0,158 |
| 5_BBXO | 8,091 | 0,130 | 12,839 | 3,02E-11 | 2327,453 | 0,277 |
| 10_BBXO | 9,107 | 0,001 | 13,363 | 3,02E-11 | 2394,460 | 0,019 |
| 25_BXO | 9,354 | 0,001 | 14,288 | 3,02E-11 | **2213,372** | **BEST** |

Table I.2: Test Root Mean Square Error (RMSE) values for each crossover on the Concrete Slump, Concrete Strength and Toxicity datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossovers** | **RMSE** | **P-value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 29,459 | 0,663 | 0,0154 | 0,088 | 136,187 | 3,83E-06 |
| 5_DBC | 30,537 | 0,137 | 0,0158 | 0,011 | 130,720 | 1,04E-04 |
| 25_DBC | 29,377 | 0,663 | 0,0159 | 0,036 | 134,180 | 6,74E-06 |
| XOBDn (min fit) | 29,858 | 0,773 | 0,0148 | 0,473 | 148,792 | 6,53E-08 |
| 5_BDBC (min fit) | 30,523 | 0,284 | 0,0160 | 0,002 | 136,012 | 7,22E-06 |
| 25_BDBC (min fit) | 29,369 | 0,520 | 0,0150 | 0,290 | 142,860 | 6,28E-06 |
| XOBDn (max size) | **27,410** | **BEST** | 0,0152 | 0,464 | 142,336 | 7,60E-07 |
| 5_BDBC (max size) | 31,085 | 0,228 | 0,0157 | 0,014 | 135,626 | 1,17E-04 |
| 25_BDBC(max size) | 29,840 | 0,340 | 0,0154 | 0,024 | 145,114 | 1,86E-06 |
| 5_DDC | 31,080 | 0,145 | 0,0168 | 1,68E-04 | 134,455 | 1,78E-04 |
| 25_DDC | 31,339 | 0,145 | 0,0157 | 0,012 | 132,796 | 1,02E-05 |
| 5_BDDC (min fit) | 29,611 | 0,579 | 0,0158 | 0,039 | 133,327 | 1,89E-04 |
| 25_BDDC (min fit) | 30,821 | 0,112 | 0,0159 | 0,021 | 136,573 | 1,58E-04 |
| 5_BDDC (max size) | 29,829 | 0,082 | 0,0166 | 1,68E-04 | 137,010 | 3,83E-05 |
| 25_BDDC (max size) | 31,175 | 0,141 | 0,0159 | 0,013 | 137,630 | 7,66E-05 |
| IDC (sem dis) | 29,443 | 0,212 | 0,0149 | 0,728 | 112,077 | 0,075 |
| IDC (fit dis) | 31,120 | 0,141 | 0,0149 | 0,923 | 117,022 | 0,019 |
| IDC (sem dis) | 28,881 | 0,807 | 0,0148 | 0,695 | 154,600 | 9,26E-09 |
| BIDC (min fit) | 28,760 | 0,569 | 0,0148 | 0,717 | 128,795 | 1,25E-04 |
| BIDC (max size) | 29,435 | 0,912 | 0,0149 | 0,695 | 134,264 | 5,61E-05 |
| BBXO | 31,082 | 0,031 | 0,0147 | 0,695 | 157,892 | 5,09E-08 |
| 5_BBXO | 29,120 | 0,641 | 0,0148 | 0,877 | **102,841** | **BEST** |
| 10_BBXO | 29,254 | 0,304 | **0,0146** | **BEST** | 104,720 | 0,695 |
| 25_BXO | 30,447 | 0,311 | 0,0150 | 0,663 | 104,541 | 0,663 |

Table I.3: Test Root Mean Square Error (RMSE) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one in terms of size. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossovers** | **RMSE** | **P-Value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 8,808 | 0,154 | 8,331 | 4,12E-06 | 2595,788 | 0,056 |
| 5_DBC | 8,405 | 0,075 | 7,339 | 0,297 | 2591,045 | 0,077 |
| 25_DBC | **7,717** | **BEST** | 9,883 | 5,57E-10 | 2592,596 | 0,072 |
| XOBDn (min fit) | 8,668 | 0,559 | 8,606 | 6,01E-08 | 2597,976 | 0,050 |
| 5_BDBC (min fit) | 8,183 | 0,819 | 8,347 | 3,52E-07 | 2593,732 | 0,064 |
| 25_BDBC (min fit) | 8,206 | 0,600 | 9,041 | 5,46E-09 | 2592,126 | 0,072 |
| XOBDn (max size) | 8,914 | 0,252 | 8,822 | 1,55E-09 | 2597,162 | 0,050 |
| 5_BDBC (max size) | 8,578 | 0,728 | 8,422 | 2,57E-07 | 2594,629 | 0,072 |
| 25_BDBC (max size) | 8,285 | 0,579 | 9,804 | 6,12E-10 | 2590,995 | 0,072 |
| 5_DDC | 9,165 | 0,041 | **7,158** | **BEST** | 2587,426 | 0,077 |
| 25_DDC | 9,122 | 0,054 | 8,746 | 5,00E-09 | 2585,362 | 0,077 |
| 5_BDDC (min fit) | 7,819 | 0,807 | 7,279 | 0,569 | 2595,922 | 0,059 |
| 25_BDDC (min fit) | 8,643 | 0,228 | 8,640 | 4,80E-07 | 2593,183 | 0,075 |
| 5_BDDC (max size) | 8,806 | 0,186 | 7,882 | 7,29E-03 | 2593,642 | 0,072 |
| 25_BDDC (max size) | 8,538 | 0,196 | 8,048 | 1,11E-04 | 2590,715 | 0,077 |
| IDC (fit imp) | 8,118 | 0,888 | 7,859 | 2,53E-04 | 2593,626 | 0,068 |
| IDC (fit dis) | 8,088 | 0,947 | 8,334 | 2,68E-06 | 2591,298 | 0,070 |
| IDC (sem dis) | 7,983 | 0,994 | 8,384 | 3,32E-06 | 2594,896 | 0,068 |
| BIDC (min fit) | 8,288 | 0,900 | 8,079 | 1,73E-06 | 2594,711 | 0,050 |
| BIDC (max size) | 8,133 | 0,579 | 8,473 | 1,20E-08 | 2592,934 | 0,050 |
| BBXO | 8,285 | 0,959 | 8,754 | 4,57E-09 | 2352,964 | 0,959 |
| 5_BBXO | 7,724 | 0,739 | 11,433 | 5,07E-10 | **2328,734** | **BEST** |
| 10_BBXO | 7,995 | 0,600 | 12,616 | 5,57E-10 | 2364,465 | 0,947 |
| 25_BXO | 9,181 | 0,031 | 14,697 | 5,07E-10 | 2368,081 | 0,900 |

Table I.4: Test Root Mean Square Error (RMSE) values for each crossover on the Concrete Slump, Concrete Strength, and Toxicity datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossovers** | **Size** | **P-Value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 543,0 | 2,74E-11 | 819,5 | 3,01E-11 | 399,0 | 2,52E-11 |
| 5_DBC | 2942,0 | 2,74E-11 | 3438,0 | 3,01E-11 | 1882,0 | 2,53E-11 |
| 25_DBC | 3648,5 | 2,74E-11 | 3991,5 | 3,01E-11 | 2844,5 | 2,53E-11 |
| XOBDn (min fit) | 205,0 | 5,46E-11 | 229,5 | 1,32E-10 | 185,0 | 2,78E-11 |
| 5_BDBC (min fit) | 866,5 | 2,74E-11 | 1437,5 | 3,01E-11 | 971,5 | 2,53E-11 |
| 25_BDBC (min fit) | 1914,0 | 2,74E-11 | 2166,0 | 3,01E-11 | 2654,0 | 2,53E-11 |
| XOBDn (max size) | 173,5 | 7,81E-11 | 217,0 | 1,47E-09 | 142,0 | 9,94E-10 |
| 5_BDBC (max size) | 3138,0 | 2,74E-11 | 4387,0 | 3,01E-11 | 3671,5 | 2,53E-11 |
| 25_BDBC (max size) | 5063,0 | 2,74E-11 | 2125,5 | 3,01E-11 | 2350,5 | 2,53E-11 |
| 5_DDC | 1944,0 | 2,74E-11 | 2921,5 | 3,01E-11 | 1510,5 | 2,53E-11 |
| 25_DDC | 4530,5 | 2,74E-11 | 5592,5 | 3,01E-11 | 5298,5 | 2,53E-11 |
| 5_BDDC (min fit) | 723,0 | 2,73E-11 | 947,0 | 3,01E-11 | 747,0 | 2,53E-11 |
| 25_BDDC (min fit) | 1735,5 | 2,74E-11 | 2105,0 | 3,01E-11 | 1477,0 | 2,53E-11 |
| 5_BDDC (max size) | 1983,0 | 2,74E-11 | 3185,5 | 3,01E-11 | 1647,0 | 2,53E-11 |
| 25_BDDC (max size) | 6265,5 | 2,74E-11 | 5588,0 | 3,01E-11 | 5948,0 | 2,53E-11 |
| IDC (fit imp) | 211,0 | 3,31E-11 | 286,5 | 2,15E-10 | 215,0 | 2,51E-11 |
| IDC (fit dis) | 241,5 | 4,96E-11 | 277,0 | 8,53E-11 | 198,0 | 2,49E-11 |
| IDC (sem dis) | 242,5 | 4,47E-11 | 216,0 | 1,40E-09 | 236,5 | 4,59E-11 |
| BIDC (min fit) | 173,5 | 2,31E-10 | 207,0 | 4,18E-10 | 162,5 | 4,56E-11 |
| BIDC (max size) | 174,5 | 5,97E-11 | 203,5 | 2,28E-08 | 131,5 | 9,10E-10 |
| BBXO | 178,0 | 5,11E-10 | 223,5 | 6,76E-08 | 121,0 | 1,19E-09 |
| 5_BBXO | 150,0 | 1,48E-07 | 174,0 | 1,53E-05 | 91,0 | 5,57E-05 |
| 10_BBXO | 102,5 | 0,149 | 165,0 | 0,001 | **75,0** | **BEST** |
| 25_BXO | **88,0** | **BEST** | 115,5 | **BEST** | 84,5 | 0,002 |

Table I.5: Individual Size (Elite Node Count) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one in terms of size. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossovers** | **Size** | **P-Value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 286,0 | 9,81E-11 | 241,0 | 2,98E-11 | 3612,5 | 3,00E-11 |
| 5_DBC | 3237,0 | 2,99E-11 | 1982,0 | 2,98E-11 | 6406,5 | 3,00E-11 |
| 25_DBC | 2673,0 | 2,99E-11 | 2525,5 | 2,98E-11 | 6773,5 | 3,00E-11 |
| XOBDn (min fit) | 115,0 | 2,33E-05 | 158,5 | 4,42E-11 | 1969 | 3,00E-11 |
| 5_BDBC (min fit) | 1269,5 | 2,99E-11 | 587,0 | 2,97E-11 | 3963 | 3,00E-11 |
| 25_BDBC (min fit) | 1357,5 | 2,99E-11 | 771,0 | 2,98E-11 | 5179,5 | 3,00E-11 |
| XOBDn (max size) | 106,0 | 1,79E-05 | 122,5 | 3,75E-10 | 1871,5 | 3,00E-11 |
| 5_BDBC (max size) | 3187,5 | 2,99E-11 | 2425,0 | 2,98E-11 | 5794,5 | 3,00E-11 |
| 25_BDBC (max size) | 1607,5 | 3,31E-11 | 2586,0 | 2,98E-11 | 8568 | 3,00E-11 |
| 5_DDC | 2019,5 | 2,99E-11 | 1250,5 | 2,98E-11 | 5801,5 | 3,00E-11 |
| 25_DDC | 3724,0 | 2,99E-11 | 1458,0 | 2,97E-11 | 7588,5 | 3,00E-11 |
| 5_BDDC (min fit) | 604,5 | 2,99E-11 | 425,0 | 2,97E-11 | 4023,5 | 3,00E-11 |
| 25_BDDC (min fit) | 2255,0 | 2,99E-11 | 834,5 | 2,98E-11 | 5335,5 | 3,00E-11 |
| 5_BDDC (max size) | 1978,0 | 2,99E-11 | 1688,0 | 2,97E-11 | 5065 | 3,00E-11 |
| 25_BDDC (max size) | 286,0 | 2,99E-11 | 3736,5 | 2,98E-11 | 8200,5 | 3,00E-11 |
| IDDC(fit imp) | 128,0 | 3,31E-08 | 142,5 | 8,41E-11 | 2562,5 | 3,00E-11 |
| IDDC(fit dis) | 124,5 | 1,41E-08 | 134,5 | 1,75E-10 | 2697,5 | 3,00E-11 |
| IDDC(sem dis) | 137,0 | 2,26E-08 | 159,5 | 9,74E-11 | 1519 | 3,00E-11 |
| BIDC (min fit) | 108,0 | 4,96E-04 | 121,0 | 2,44E-10 | 1720,5 | 3,00E-11 |
| BIDC (max size) | 114,0 | 3,67E-06 | 133,0 | 4,69E-09 | 1930,5 | 3,00E-11 |
| BBXO | 132,0 | 1,47E-05 | 113,5 | 9,74E-07 | 1261 | 3,00E-11 |
| 5_BBXO | 125,0 | 8,12E-07 | 91,5 | 8,05E-04 | 701,5 | 3,86E-11 |
| 10_BBXO | 86,5 | 0,105 | 85,5 | 3,39E-03 | 478,5 | 0,028 |
| 25_BXO | **75,0** | **BEST** | **68,0** | **BEST** | **414** | **BEST** |

Table I.6: Individual Size (Elite Node Count) values for each crossover on the Concrete Slump, Concrete Strength, and Toxicity datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one in terms of size. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossovers** | **Size** | **P-Value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 4090,0 | 3,01E-11 | 968,0 | 3,01E-11 | 4585,5 | 3,02E-11 |
| 5_DBC | 6505,5 | 3,02E-11 | 3337,0 | 3,01E-11 | 7095,5 | 3,02E-11 |
| 25_DBC | 7227,0 | 3,02E-11 | 3725,0 | 3,01E-11 | 8056,5 | 3,02E-11 |
| XOBDn (min fit) | 2154,0 | 3,00E-11 | 299,5 | 4,49E-11 | 2475,5 | 3,02E-11 |
| 5_BDBC (min fit) | 5046,5 | 3,02E-11 | 1346,0 | 3,01E-11 | 6029,5 | 3,02E-11 |
| 25_BDBC (min fit) | 5785,0 | 3,01E-11 | 3993,5 | 3,01E-11 | 6256,5 | 3,02E-11 |
| XOBDn (max size) | 2100,5 | 3,02E-11 | 267,5 | 9,88E-11 | 2493,5 | 3,02E-11 |
| 5_BDBC (max size) | 5986,5 | 3,02E-11 | 3368,0 | 3,01E-11 | 6897 | 3,02E-11 |
| 25_BDBC (max size) | 6800,5 | 3,01E-11 | 2493,5 | 3,01E-11 | 8189,5 | 3,02E-11 |
| 5_DDC | 5989,0 | 3,01E-11 | 3306,5 | 3,01E-11 | 6900 | 3,02E-11 |
| 25_DDC | 7275,5 | 3,02E-11 | 4853,5 | 3,01E-11 | 8497,5 | 3,02E-11 |
| 5_BDDC (min fit) | 4519,0 | 3,02E-11 | 898,0 | 3,01E-11 | 5435,5 | 3,02E-11 |
| 25_BDDC (min fit) | 5735,0 | 3,01E-11 | 2681,5 | 3,01E-11 | 6588 | 3,02E-11 |
| 5_BDDC (max size) | 5170,5 | 3,01E-11 | 3014,0 | 3,01E-11 | 6073,5 | 3,02E-11 |
| 25_BDDC (max size) | 7352,0 | 3,02E-11 | 5080,5 | 3,01E-11 | 8237,5 | 3,02E-11 |
| IDDC(fit imp) | 4698,0 | 3,01E-11 | 348,5 | 5,47E-11 | 6284 | 3,02E-11 |
| IDDC(fit dis) | 4513,5 | 3,01E-11 | 315,0 | 5,21E-11 | 6276 | 3,02E-11 |
| IDDC(sem dis) | 2081,0 | 3,02E-11 | 319,0 | 3,67E-11 | 2252,5 | 3,02E-11 |
| BIDC (min fit) | 2789,5 | 3,02E-11 | 270,0 | 2,48E-10 | 3264 | 3,02E-11 |
| BIDC (max size) | 2628,0 | 3,02E-11 | 236,5 | 2,86E-10 | 3289,5 | 3,02E-11 |
| BBXO | 4661,5 | 3,02E-11 | 312,5 | 1,77E-09 | 1871,5 | 3,02E-11 |
| 5_BBXO | 2392,5 | 3,02E-11 | 200,5 | 1,36E-07 | 1572 | 3,02E-11 |
| 10_BBXO | 903,5 | 1,77E-10 | 159,0 | 0,002 | 678 | 1,33E-10 |
| 25_BXO | **441,0** | **BEST** | **122,5** | **BEST** | **341,5** | **BEST** |

Table I.7: Individual Size (Elite Node Count) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one in terms of size. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

| Datasets | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossovers** | **Size** | **P-Value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 2640,5 | 3,02E-11 | 3165,5 | 3,02E-11 | 6301,0 | 3,01E-11 |
| 5_DBC | 6416,5 | 3,02E-11 | 5663,5 | 3,02E-11 | 10825,5 | 3,00E-11 |
| 25_DBC | 8821,5 | 3,02E-11 | 6717,0 | 3,02E-11 | 10935,0 | 2,78E-11 |
| XOBDn (min fit) | 1064,0 | 3,02E-11 | 1737,0 | 3,02E-11 | 2988,5 | 3,01E-11 |
| 5_BDBC (min fit) | 3380,5 | 3,02E-11 | 3349,5 | 3,02E-11 | 9457,0 | 3,01E-11 |
| 25_BDBC (min fit) | 5431,5 | 3,02E-11 | 5189,5 | 3,02E-11 | 10937,0 | 1,94E-11 |
| XOBDn (max size) | 1194,5 | 3,02E-11 | 1790,0 | 3,02E-11 | 3018,0 | 2,98E-11 |
| 5_BDBC (max size) | 6258,5 | 3,02E-11 | 5891,5 | 3,02E-11 | 10364,5 | 3,01E-11 |
| 25_BDBC (max size) | 8355,5 | 3,02E-11 | 8313,0 | 3,02E-11 | 10888,5 | 3,00E-11 |
| 5_DDC | 5854,0 | 3,02E-11 | 5381,0 | 3,02E-11 | 10358,0 | 3,01E-11 |
| 25_DDC | 10237,0 | 3,02E-11 | 8113,5 | 3,02E-11 | 10926,0 | 2,75E-11 |
| 5_BDDC (min fit) | 2690,5 | 3,02E-11 | 2712,0 | 3,02E-11 | 8015,0 | 3,00E-11 |
| 25_BDDC (min fit) | 5092,5 | 3,02E-11 | 4454,0 | 3,02E-11 | 10893,0 | 2,89E-11 |
| 5_BDDC (max size) | 5046,0 | 3,02E-11 | 4993,5 | 3,02E-11 | 8242,0 | 2,98E-11 |
| 25_BDDC (max size) | 10284,5 | 3,02E-11 | 8613,5 | 3,02E-11 | 10836,5 | 3,00E-11 |
| IDDC(fit imp) | 1044,5 | 3,02E-11 | 1545,0 | 3,02E-11 | 6530,0 | 3,01E-11 |
| IDDC(fit dis) | 988,0 | 3,02E-11 | 1615,0 | 3,02E-11 | 6468,5 | 3,01E-11 |
| IDC (sem dis) | 869,5 | 3,02E-11 | 1540,5 | 3,02E-11 | 6455,0 | 3,01E-11 |
| BIDC (min fit) | 914,0 | 3,02E-11 | 1291,0 | 3,33E-11 | 3509,0 | 3,00E-11 |
| BIDC (max size) | 989,5 | 3,02E-11 | 1552,0 | 3,02E-11 | 3458,0 | 2,99E-11 |
| BBXO | 1275,5 | 3,02E-11 | 1452,5 | 3,02E-11 | 3972,5 | 3,01E-11 |
| 5_BBXO | 977,5 | 4,08E-11 | 1256,5 | 4,07E-11 | 2322,0 | 3,01E-11 |
| 10_BBXO | 922,5 | 3,02E-11 | 956,0 | 7,38E-10 | 1069,0 | 7,37E-11 |
| 25_BXO | **402,5** | **BEST** | **626,0** | **BEST** | **568,0** | **BEST** |

Table I.8: Individual Size (Elite Node Count) values for each crossover on the Concrete Slump, Concrete Strength, and Toxicity datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test compare each crossover method to the best-performing one in terms of size. The best-performing crossover is indicated in **bold** and labeled **BEST**. Following ranked crossover operators not significantly different from the best ($p \geq 0.01$) are highlighted in green.

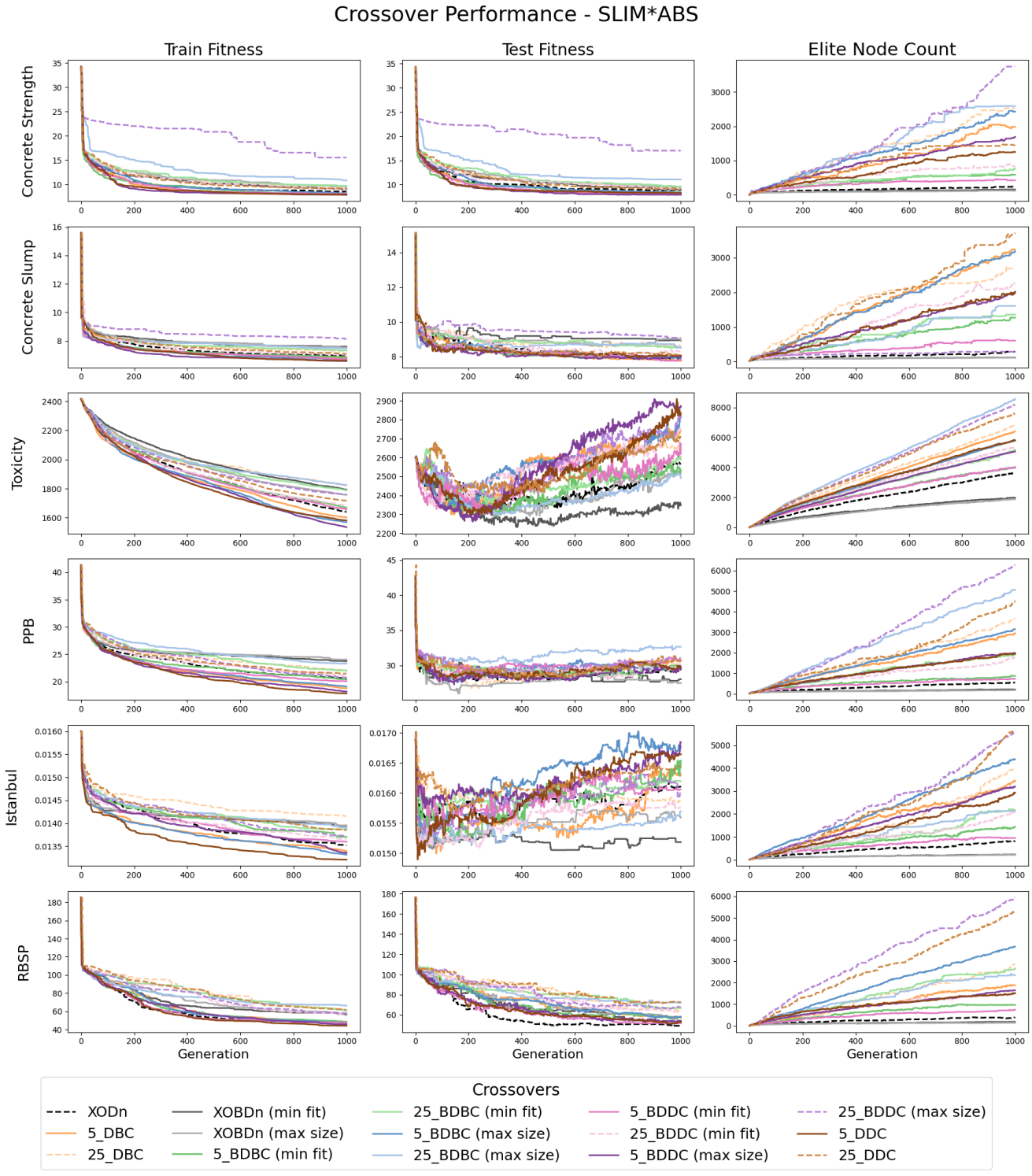


Figure I.1: **Random Block Crossover** performance across datasets for the **SLIM*ABS** variant. Lines show the evolution of the training fitness, testing fitness, and elite node count across generations, with median values across 30 runs for each crossover operator.

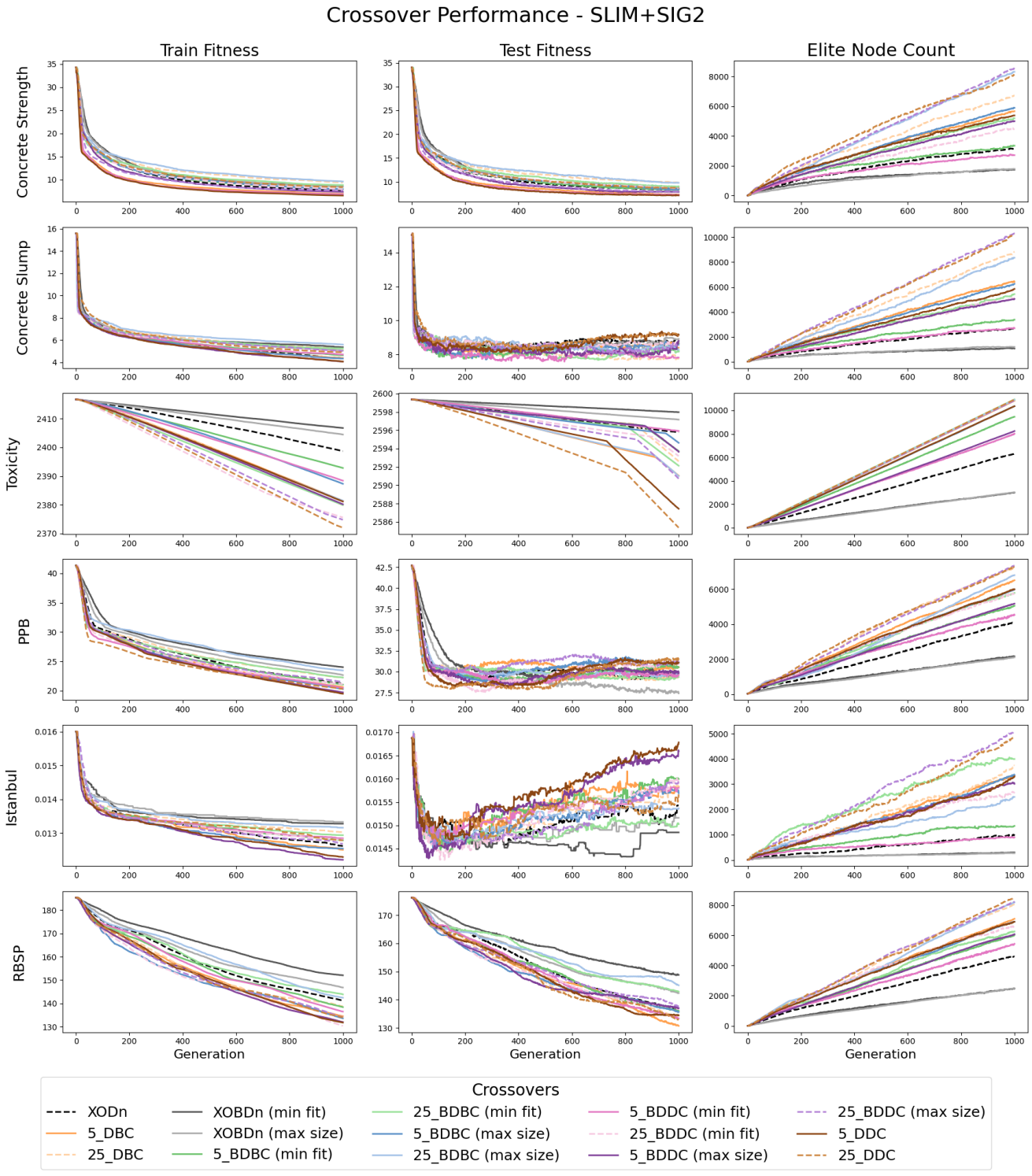


Figure I.2: **Random Block Crossover** operators performance across datasets for the **SLIM+SIG2** variant. Lines show the evolution of the training fitness, testing fitness, and elite node count across generations, with median values across 30 runs for each crossover operator.

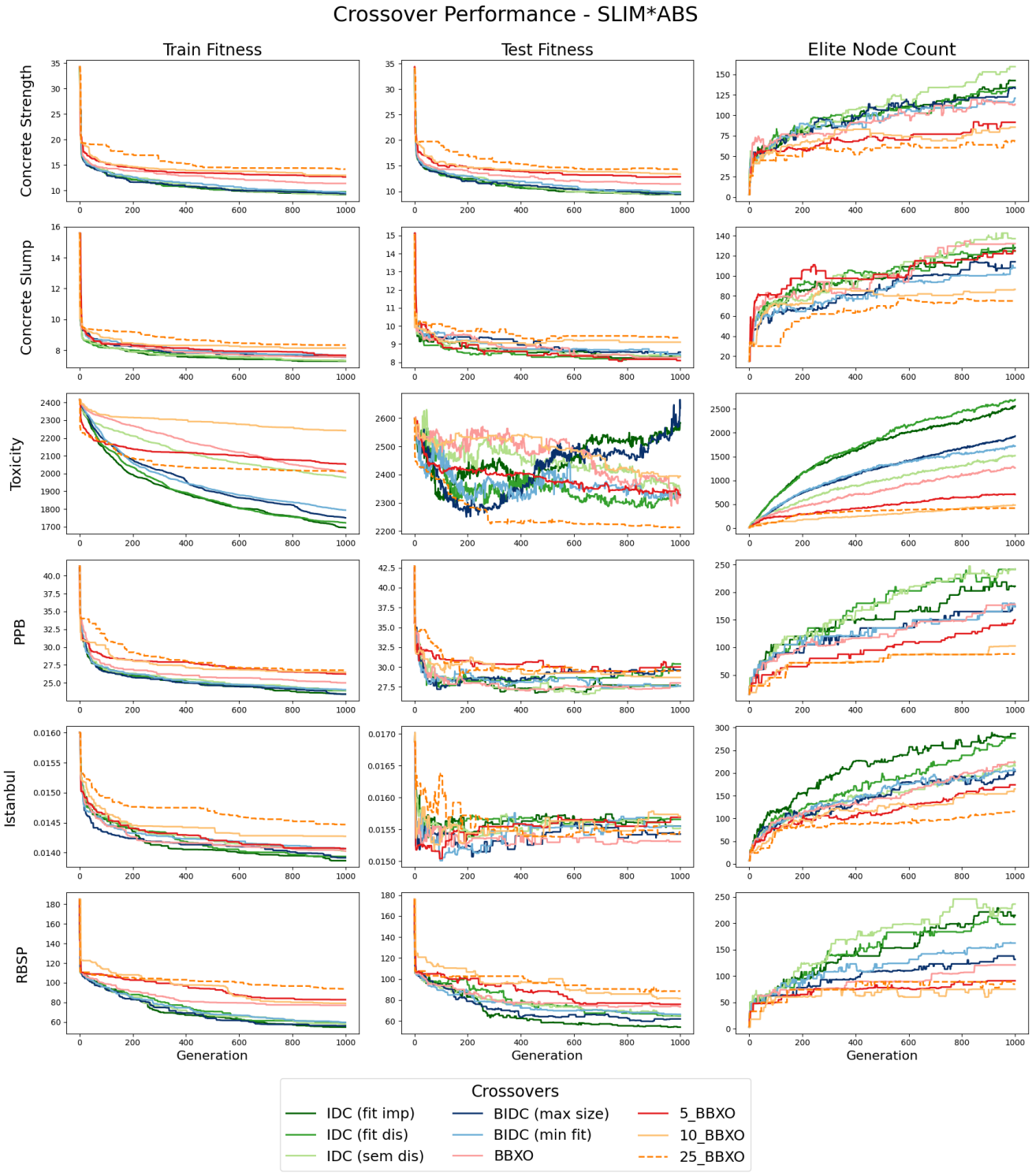


Figure I.3: **Startegic Block Crossover** operators performance across datasets for the **SLIM*ABS** variant. Lines show the evolution of the training fitness, testing fitness, and elite node count across generations, with median values across 30 runs for each crossover operator.

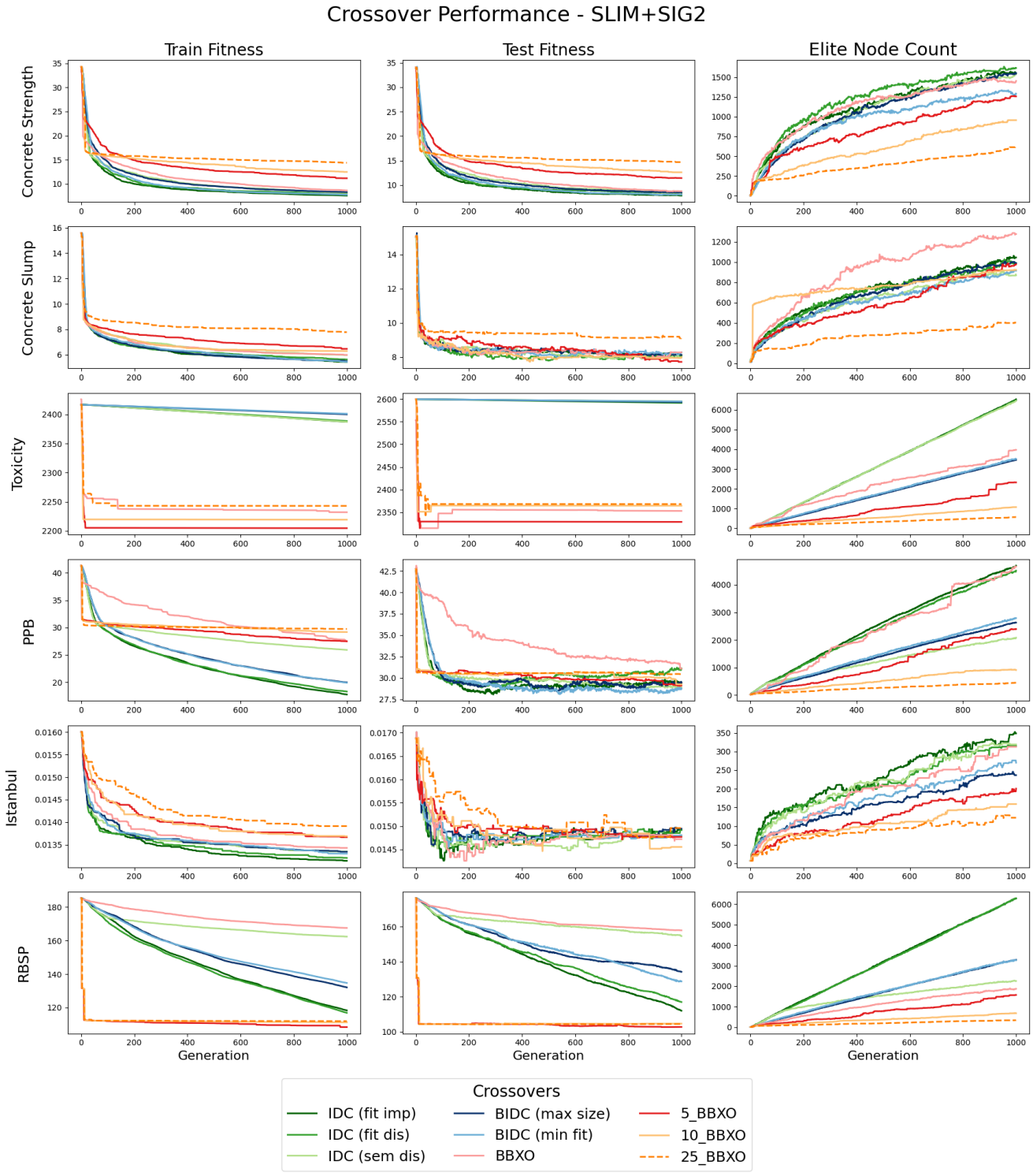


Figure I.4: **Strategic Block Crossover** operators performance across datasets for the **SLIM+SIG2** variant. Lines show the evolution of the training fitness, testing fitness, and elite node count across generations, with median values across 30 runs for each crossover operator.

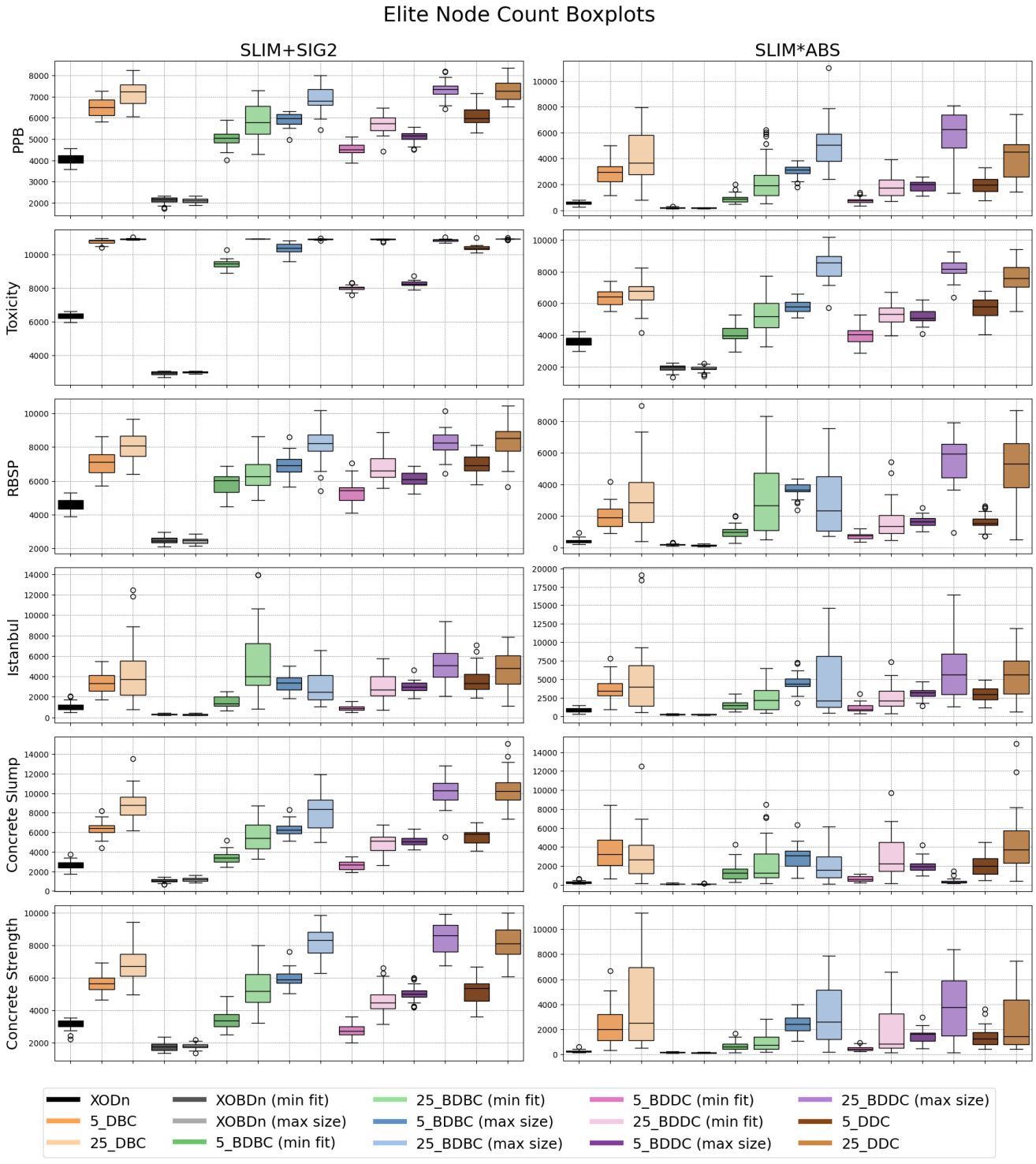


Figure I.5: **Random Block Crossover** operators *elite node counts* across datasets. The boxplots display the values of individual size in the final generation, across 30 runs for each crossover operator.

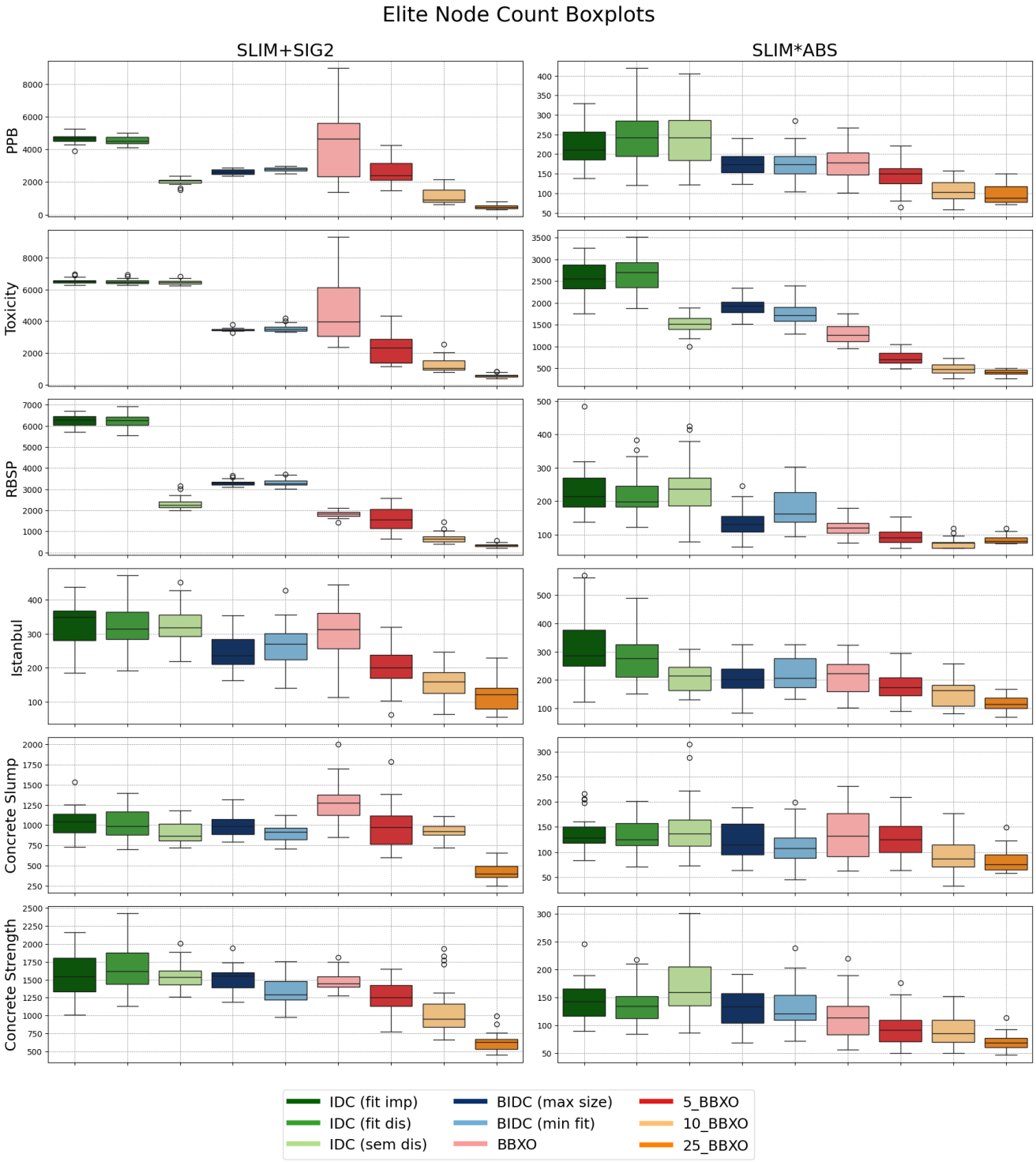


Figure I.6: **Strategic Block Crossover** operators *elite node counts* across datasets. The boxplots display the values of individual size in the final generation, across 30 runs for each crossover operator.

| Dataset | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossover** | **RMSE** | **P-value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 29,275 | 0,465 | 0,016 | 0,191 | 49,263 | **1,19E-06** |
| XOBDn (min fit) | 28,012 | 0,761 | 0,015 | **0,007** | 53,428 | **4,66E-03** |
| XOBDn (max size) | 27,468 | 0,058 | 0,016 | 0,213 | 54,390 | **7,06E-05** |
| IDC (fit imp) | 27,632 | 0,855 | 0,016 | 0,685 | 54,129 | **9,90E-05** |
| IDC (fit dis) | 30,399 | 0,626 | 0,016 | 0,808 | 64,878 | 0,404 |
| IDC (sem dis) | 27,657 | 0,146 | 0,016 | 0,670 | 65,615 | 0,416 |
| BIDC (min fit) | 27,626 | 0,404 | 0,016 | 0,339 | 66,374 | 0,080 |
| BIDC (max size) | 29,614 | 0,299 | 0,015 | 0,050 | 61,963 | **0,003** |
| BBXO | 28,014 | 0,077 | 0,015 | 0,119 | 73,511 | 0,124 |
| 5_BBXO | 30,017 | 0,198 | 0,016 | 0,061 | 75,925 | 2,37E-05 |
| 10_BBXO | 28,707 | 0,641 | 0,016 | 0,096 | 81,424 | 0,003 |
| 25_BBXO | 29,522 | 0,730 | 0,015 | 0,903 | 88,514 | 3,54E-08 |
| **SLIM** | 29,312 | | 0,016 | | 66,346 | |

Table I.9: **Test Fitness** (RMSE) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossover** | **RMSE** | **P-Value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 7,903 | **0,007** | 8,853 | **6,29E-05** | 2568,961 | 9,00E-06 |
| XOBDn (min fit) | 8,944 | 0,824 | 8,973 | 0,013 | 2347,260 | 0,084 |
| XOBDn (max size) | 8,889 | 0,543 | 9,499 | 0,839 | 2523,960 | 0,003 |
| IDC (fit imp) | 8,333 | 0,088 | 9,342 | 0,033 | 2583,744 | 0,001 |
| IDC (fit dis) | 8,285 | 0,119 | 9,849 | 0,855 | 2328,190 | 0,012 |
| IDC (sem dis) | 8,336 | 0,080 | 9,343 | 0,124 | 2342,122 | 0,035 |
| BIDC (min fit) | 8,407 | 0,871 | 9,641 | 0,584 | 2323,849 | 0,002 |
| BIDC (max size) | 8,557 | 0,626 | 9,440 | 0,428 | 2635,632 | 0,001 |
| BBXO | 8,255 | 0,328 | 11,426 | 0,001 | 2319,648 | 0,839 |
| 5_BBXO | 8,091 | 0,529 | 12,839 | 2,61E-08 | 2327,453 | 0,529 |
| 10_BBXO | 9,107 | 0,020 | 13,363 | 3,73E-09 | 2394,460 | 0,013 |
| 25_BBXO | 9,354 | 0,003 | 14,288 | 3,73E-09 | 2213,372 | **3,45E-04** |
| **SLIM** | 8,638 | | 9,655 | | 2322,873 | |

Table I.10: **Test Fitness** (RMSE) values for each crossover on the Concrete Slump, Concrete Strength and Toxicity datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossover** | **RMSE** | **P-value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 29,459 | 0,652 | 0,015 | 0,093 | 136,187 | **2,53E-04** |
| XOBDn (min fit) | 29,858 | 0,589 | 0,015 | 0,464 | 148,792 | 0,047 |
| XOBDn (max size) | 27,410 | 0,900 | 0,015 | 0,569 | 142,336 | **9,52E-04** |
| IDC (fit imp) | 29,443 | 0,223 | 0,015 | 0,877 | 112,077 | **1,47E-07** |
| IDC (fit dis) | 31,120 | 0,088 | 0,015 | 0,912 | 117,022 | **9,06E-08** |
| IDC (sem dis) | 28,881 | 0,706 | 0,015 | 0,796 | 154,600 | 0,540 |
| BIDC (min fit) | 28,760 | 0,579 | 0,015 | 0,807 | 128,795 | **6,28E-06** |
| BIDC (max size) | 29,435 | 0,982 | 0,015 | 0,739 | 134,264 | **2,28E-05** |
| BBXO | 31,082 | 0,021 | 0,015 | 0,751 | 157,892 | 0,971 |
| 5_BBXO | 29,120 | 0,610 | 0,015 | 1,000 | 102,841 | **5,46E-09** |
| 10_BBXO | 29,254 | 0,212 | 0,015 | 0,971 | 104,720 | **8,48E-09** |
| 25_BBXO | 30,447 | 0,246 | 0,015 | 0,706 | 104,541 | **1,01E-08** |
| **SLIM** | 28,740 | | 0,014 | | 156,627 | |

Table I.11: **Test Fitness** (RMSE) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossover** | **RMSE** | **P-Value** | **RMSE** | **P-Value** | **RMSE** | **P-Value** |
| XODn | 8,808 | 0,264 | 8,331 | 0,030 | 2595,788 | 0,751 |
| XOBDn (min fit) | 8,668 | 0,610 | 8,606 | 0,728 | 2597,976 | 0,830 |
| XOBDn (max size) | 8,914 | 0,404 | 8,822 | 0,429 | 2597,162 | 0,796 |
| IDC (fit imp) | 8,118 | 0,589 | 7,859 | **1,11E-04** | 2593,626 | 0,706 |
| IDC (fit dis) | 8,088 | 0,641 | 8,334 | 0,109 | 2591,298 | 0,717 |
| IDC (sem dis) | 7,983 | 0,600 | 8,384 | 0,112 | 2594,896 | 0,739 |
| BIDC (min fit) | 8,288 | 0,641 | 8,079 | **1,06E-03** | 2594,711 | 0,762 |
| BIDC (max size) | 8,133 | 0,912 | 8,473 | 0,631 | 2592,934 | 0,773 |
| BBXO | 8,285 | 0,830 | 8,754 | 0,935 | 2352,964 | 0,047 |
| 5_BBXO | 7,724 | 0,473 | 11,433 | 8,10E-10 | 2328,734 | 0,050 |
| 10_BBXO | 7,995 | 0,363 | 12,616 | 1,69E-09 | 2364,465 | 0,044 |
| 25_BBXO | 9,181 | 0,064 | 14,697 | 5,07E-10 | 2368,081 | 0,057 |
| **SLIM** | 8,358 | | 8,785 | | 2598,406 | |

Table I.12: **Test Fitness** (RMSE) values for each crossover on the Concrete Slump, Concrete Strength and Toxicity datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossover** | **Size** | **P-value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 543,0 | 1,86E-09 | 819,5 | 3,73E-09 | 399,0 | 1,30E-08 |
| XOBDn (min fit) | 205,0 | 0,787 | 229,5 | 0,152 | 185,0 | 0,055 |
| XOBDn (max size) | 173,5 | **0,006** | 217,0 | **0,003** | 142,0 | **0,005** |
| IDC (fit imp) | 211,0 | 0,107 | 286,5 | 0,016 | 215,0 | 4,06E-04 |
| IDC (fit dis) | 241,5 | 0,003 | 277,0 | 0,221 | 198,0 | 0,003 |
| IDC (sem dis) | 242,5 | 0,026 | 216,0 | **0,001** | 236,5 | 4,97E-05 |
| BIDC (min fit) | 173,5 | 0,064 | 207,0 | 0,027 | 162,5 | 0,556 |
| BIDC (max size) | 174,5 | 0,045 | 203,5 | **2,56E-04** | 131,5 | **0,003** |
| BBXO | 178,0 | 0,022 | 223,5 | **0,002** | 121,0 | **7,06E-05** |
| 5_BBXO | 150,0 | **2,09E-04** | 174,0 | **1,60E-05** | 91,0 | **1,86E-08** |
| 10_BBXO | 102,5 | **6,15E-08** | 165,0 | **1,30E-07** | 75,0 | **3,73E-09** |
| 25_BBXO | 88,0 | **1,86E-09** | 115,5 | **1,86E-09** | 84,5 | **2,61E-08** |
| **SLIM** | 198,0 | | 273,5 | | 170,0 | |

Table I.13: **Individual Size** (Elite Node Count) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossover** | **Size** | **P-Value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 286,0 | 5,72E-07 | 241,0 | 1,02E-07 | 3612,5 | 1,86E-09 |
| XOBDn (min fit) | 115,0 | 0,058 | 158,5 | 0,135 | 1969,0 | 0,855 |
| XOBDn (max size) | 106,0 | **0,001** | 122,5 | 0,073 | 1871,5 | 0,096 |
| IDC (fit imp) | 128,0 | 0,318 | 142,5 | 0,903 | 2562,5 | 1,30E-08 |
| IDC (fit dis) | 124,5 | 0,503 | 134,5 | 0,315 | 2697,5 | 1,30E-08 |
| IDC (sem dis) | 137,0 | 0,360 | 159,5 | 0,229 | 1519,0 | **1,86E-09** |
| BIDC (min fit) | 108,0 | **0,002** | 121,0 | 0,198 | 1720,5 | **4,60E-04** |
| BIDC (max size) | 114,0 | 0,031 | 133,0 | 0,070 | 1930,5 | 0,477 |
| BBXO | 132,0 | 0,584 | 113,5 | 0,022 | 1261,0 | **1,86E-09** |
| 5_BBXO | 125,0 | 0,280 | 91,5 | **6,29E-05** | 701,5 | **1,86E-09** |
| 10_BBXO | 86,5 | **1,11E-04** | 85,5 | **5,77E-05** | 478,5 | **1,86E-09** |
| 25_BBXO | 75,0 | **1,19E-06** | 68,0 | **1,86E-09** | 414,0 | **1,86E-09** |
| **SLIM** | 141,0 | | 140,0 | | 1957,0 | |

Table I.14: **Individual Size** (Elite Node Count) values for each crossover on the Concrete Slump, Concrete strength and Toxicity datasets, for the SLIM*ABS variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | PPB | | Istanbul | | RBSP | |
|---|---|---|---|---|---|---|
| **Crossover** | **Size** | **P-value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 4090,0 | 3,02E-11 | 968,0 | 4,97E-11 | 4585,5 | 3,02E-11 |
| XOBDn (min fit) | 2154,0 | **8,92E-05** | 299,5 | 0,121 | 2475,5 | 0,271 |
| XOBDn (max size) | 2100,5 | **6,74E-06** | 267,5 | 0,019 | 2493,5 | 0,492 |
| IDC (fit imp) | 4698,0 | 3,02E-11 | 348,5 | 0,959 | 6284,0 | 3,02E-11 |
| IDC (fit dis) | 4513,5 | 3,02E-11 | 315,0 | 0,773 | 6276,0 | 3,02E-11 |
| IDC (sem dis) | 2081,0 | **9,18E-07** | 319,0 | 0,918 | 2252,5 | **0,006** |
| BIDC (min fit) | 2789,5 | 6,07E-11 | 270,0 | **0,002** | 3264,0 | 3,33E-11 |
| BIDC (max size) | 2628,0 | 9,67E-09 | 236,5 | **2,86E-05** | 3289,5 | 3,02E-11 |
| BBXO | 4661,5 | 1,06E-03 | 312,5 | 0,387 | 1871,5 | **1,17E-05** |
| 5_BBXO | 2392,5 | 0,297 | 200,5 | **1,11E-07** | 1572,0 | **5,53E-08** |
| 10_BBXO | 903,5 | **5,49E-11** | 159,0 | **1,20E-10** | 678,0 | **3,02E-11** |
| 25_BBXO | 441,0 | **3,02E-11** | 122,5 | **4,49E-11** | 341,5 | **3,02E-11** |
| **SLIM** | 2306,0 | | 324,0 | | 2434,0 | |

Table I.15: **Individual Size** (Elite Node Count) values for each crossover on the PPB, Istanbul, and RBSP datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

| Dataset | Concrete Slump | | Concrete Strength | | Toxicity | |
|---|---|---|---|---|---|---|
| **Crossover** | **Size** | **P-Value** | **Size** | **P-Value** | **Size** | **P-Value** |
| XODn | 2640,5 | 3,02E-11 | 3165,5 | 1,09E-10 | 6301,0 | 3,02E-11 |
| XOBDn (min fit) | 1064,0 | **7,90E-05** | 1737,0 | **6,28E-07** | 2988,5 | **3,44E-06** |
| XOBDn (max size) | 1194,5 | 0,067 | 1790,0 | **2,47E-07** | 3018,0 | **1,17E-04** |
| IDC (fit imp) | 1044,5 | **2,89E-07** | 1545,0 | **3,20E-09** | 6530,0 | 3,01E-11 |
| IDC (fit dis) | 988,0 | **3,70E-06** | 1615,0 | **1,03E-06** | 6468,5 | 3,02E-11 |
| IDC (sem dis) | 869,5 | **3,01E-10** | 1540,5 | **2,15E-10** | 6455,0 | 3,02E-11 |
| BIDC (min fit) | 914,0 | **1,33E-10** | 1291,0 | **3,33E-11** | 3509,0 | 1,28E-09 |
| BIDC (max size) | 989,5 | **3,80E-08** | 1552,0 | **9,91E-11** | 3458,0 | 1,28E-09 |
| BBXO | 1275,5 | 0,751 | 1452,5 | **3,47E-10** | 3972,5 | 0,015 |
| 5_BBXO | 977,5 | **5,60E-07** | 1256,5 | **3,02E-11** | 2322,0 | **1,49E-06** |
| 10_BBXO | 922,5 | **8,15E-11** | 956,0 | **8,99E-11** | 1069,0 | **3,02E-11** |
| 25_BBXO | 402,5 | **3,02E-11** | 626,0 | **3,02E-11** | 568,0 | **3,01E-11** |
| **SLIM** | 1260,5 | | 2201,0 | | 3143,0 | |

Table I.16: **Individual Size** (Elite Node Count) values for each crossover on the Concrete Slump, Concrete Strength and Toxisity datasets, for the SLIM+SIG2 variant. P-values from the Mann-Whitney U test ($p \geq 0.01$) compare each crossover method to the corresponding standard SLIM_GSGP version in terms of test fitness. The crossover that are statistically better than standard SLIM is indicated in **bold**.

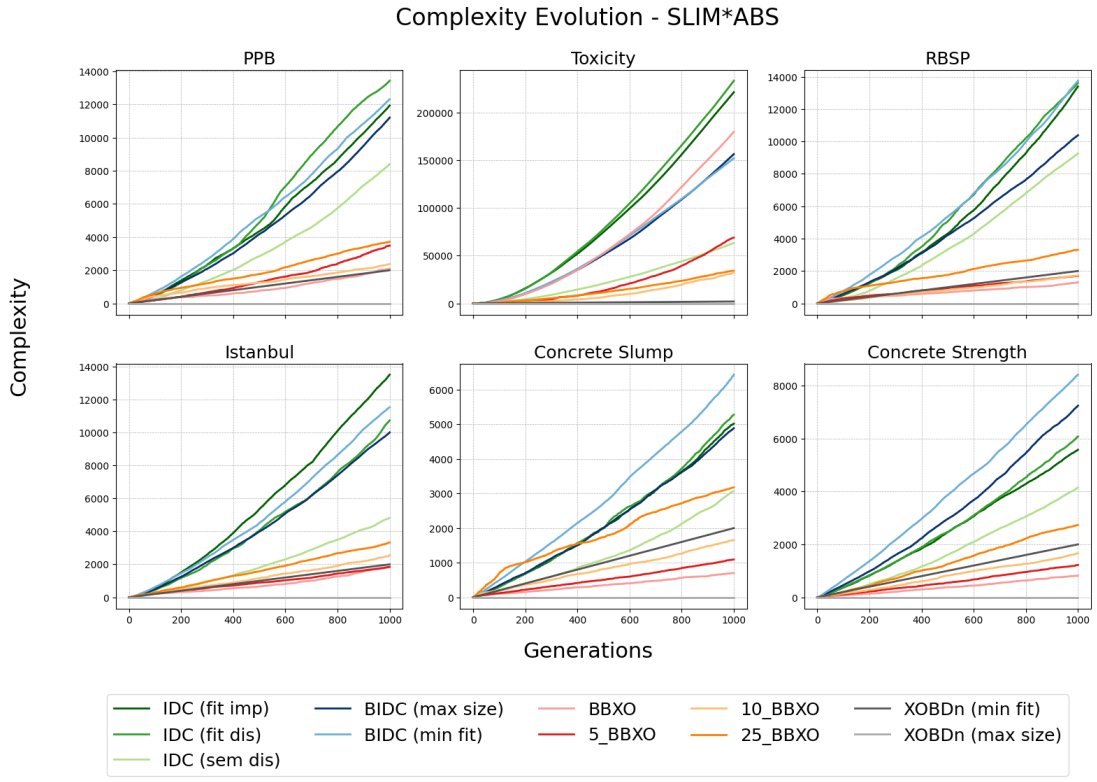


Figure I.7: Complexity Study of **Strategic Block Crossovers** and **XOBDn**. Lines show the evolution of the training fitness, testing fitness, and elite node count for SLIM*ABS, across generations, with median values across 30 runs for each crossover operator.

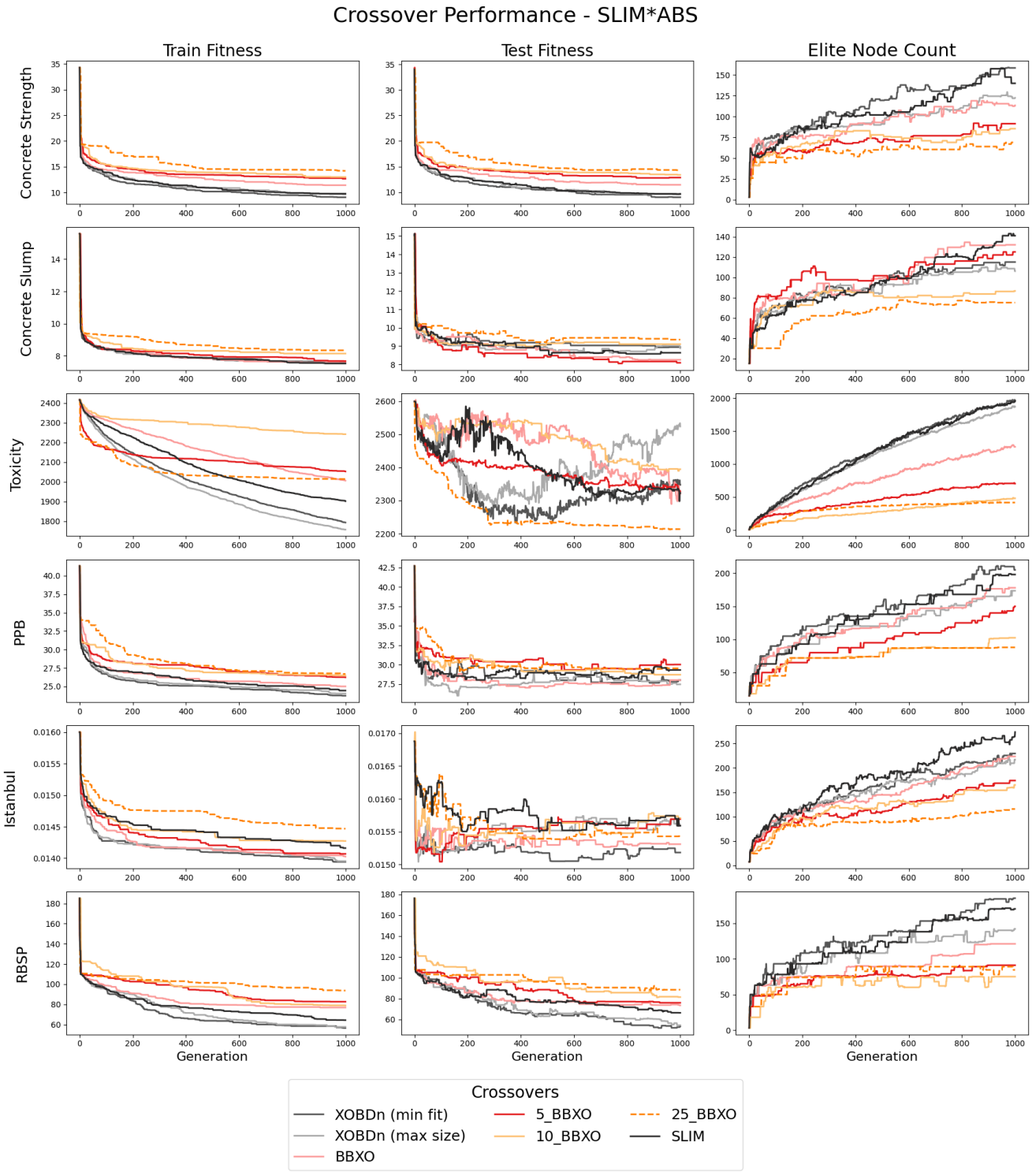


Figure I.8: **BBXO** and **XOBDn** operators performance across datasets for the **SLIM*ABS** variant. Lines show the evolution of the training fitness, testing fitness, and elite node count across generations, with median values across 30 runs for each crossover operator and the corresponding mutation-only SLIM version.

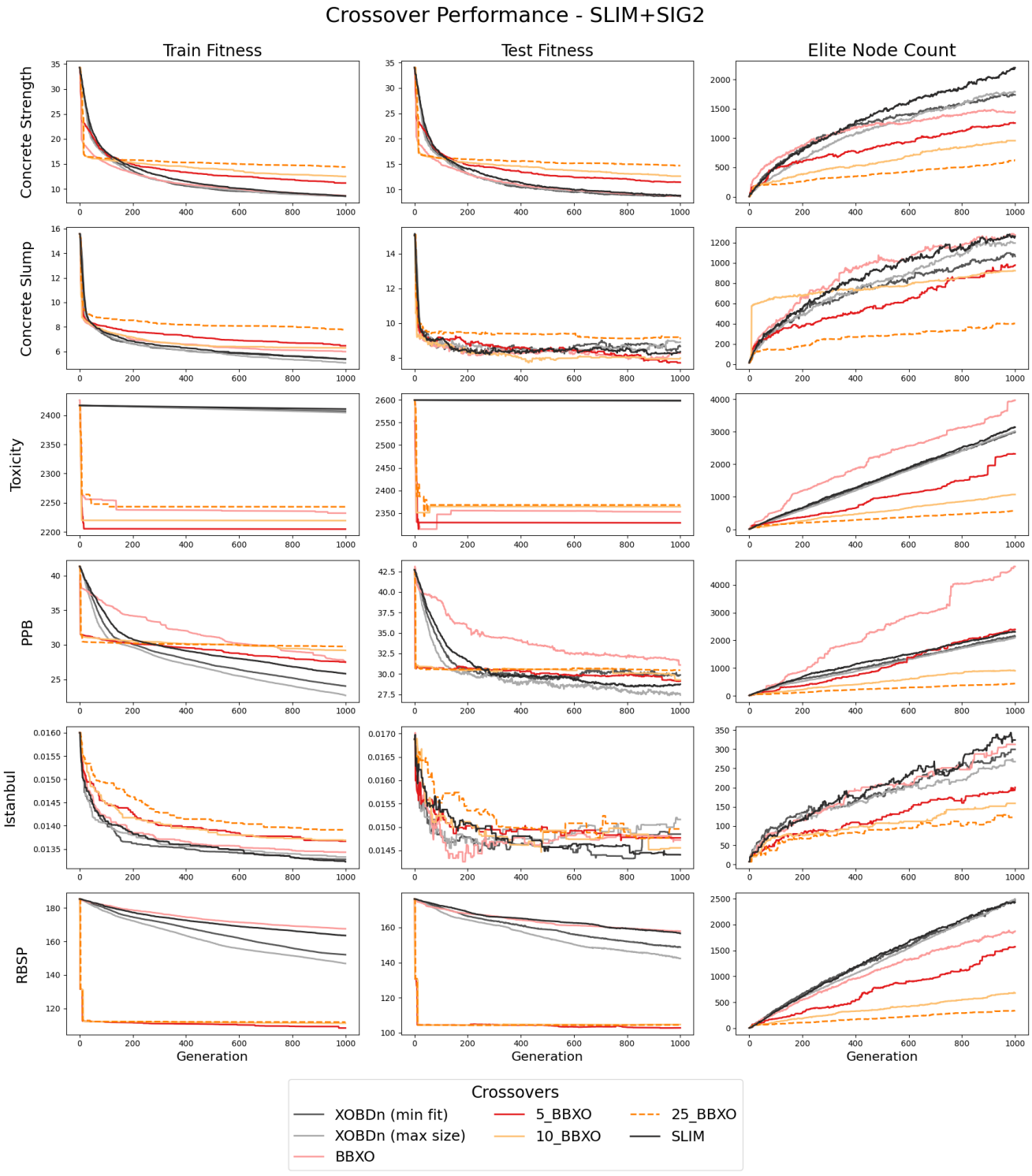


Figure I.9: **BBXO** and **XOBDn** operators performance across datasets for the **SLIM+SIG2** variant. Lines show the evolution of the training fitness, testing fitness, and elite node count across generations, with median values across 30 runs for each crossover operator and the corresponding mutation-only SLIM version.

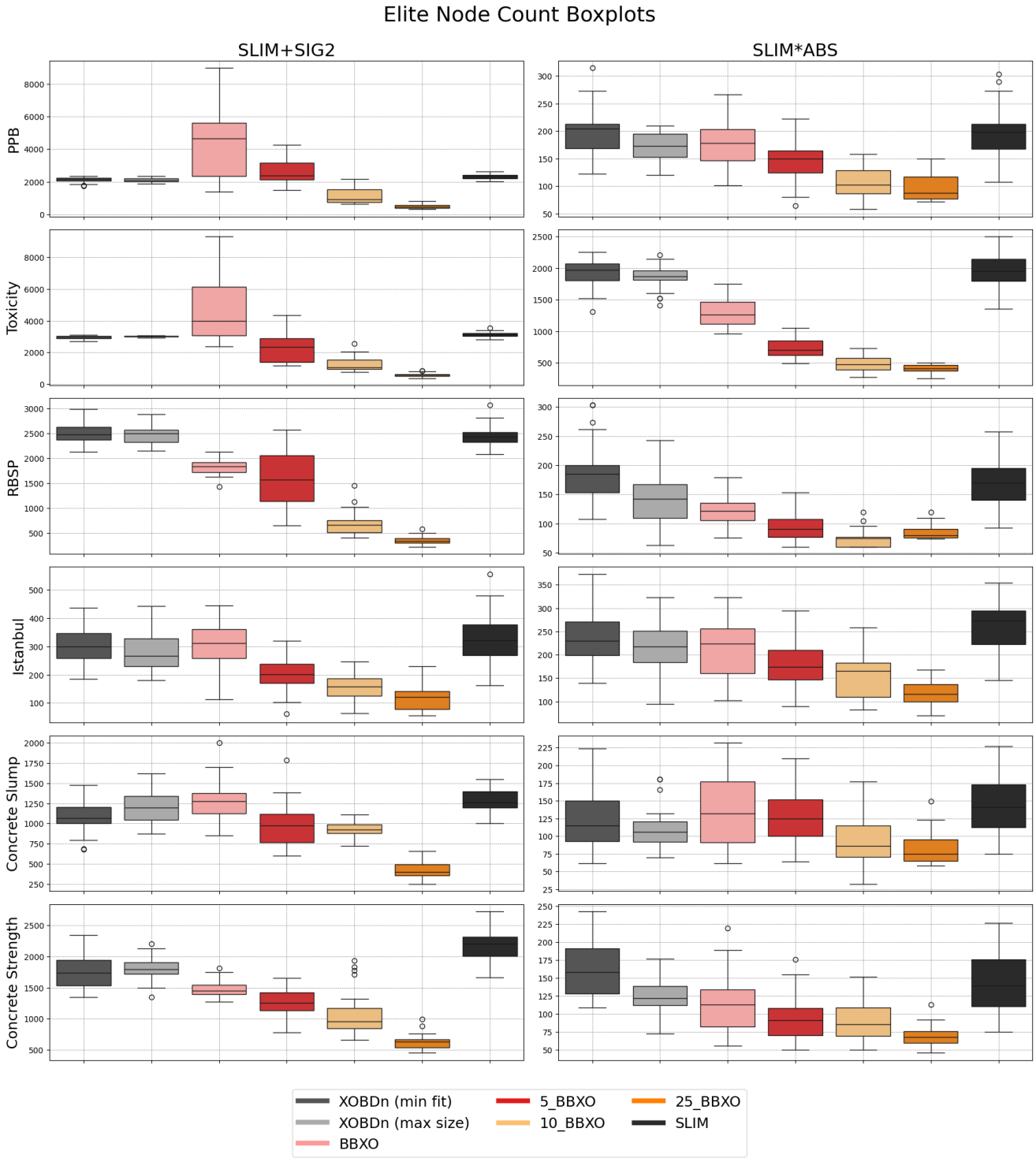


Figure I.10: **BBXO** and **XOBDn** operators *elite node counts* across datasets. The boxplots display the values of individual size in the final generation, across 30 runs for each crossover operator and the corresponding mutation-only SLIM version.


**NOVA Information Management School**
**Instituto Superior de Estatística e Gestão de Informação**

Universidade NOVA de Lisboa