Multimodal Adaptive Graph Evolution

Camilo de la Torre, Kévin Cortacero, Dennis Wilson, Sylvain Cussat-Blanc

Multimodal Adaptive Graph Evolution

Camilo De La Torre
camilo.de-la-torre@ut-capitole.fr
Université Toulouse Capitole - IRIT CNRS UMR5505
Toulouse, France

Kevin Cortacero
kevin.cortacero@inserm.fr
CRCT - INSERM
Toulouse, France

Dennis G Wilson
dennis.wilson@isae-supaero.fr
ISAE-Supaero, University of Toulouse
Toulouse, France

Sylvain Cussat-Blanc
sylvain.cussat-blanc@irit.fr
Université Toulouse Capitole - IRIT CNRS UMR5505
Toulouse, France

ABSTRACT

The problem of program synthesis involves automatically finding a function based on evaluation criteria, like matching input-output pairs. While Cartesian Genetic Programming (CGP) has excelled in various function synthesis tasks, it has primarily been limited to single data types, hindering its applicability to diverse data. Mixed-Type CGP, proposed in 2012, aimed to address this limitation but faced challenges due to search space limitations and complexity in building function libraries. In this study, we introduce Multimodal Adaptive Graph Evolution (MAGE), a generalized CGP extension that integrates functions of different data types by grouping them accordingly and imposing mutation constraints based on type. Through comparisons with standard CGP and Mixed-Type CGP on Program Synthesis Benchmark and image classification tasks, we demonstrate that MAGE's representation and mutation constraints facilitate the search for multimodal functions.

CCS CONCEPTS

• **Software and its engineering → Genetic programming**; • **Theory of computation → Evolutionary algorithms**.

KEYWORDS

Genetic Programming, Evolutionary Computation, Symbolic Regression

1 INTRODUCTION

In modern scientific data analysis, the integration of diverse data types, ranging from numerical integers to textual strings and complex multi-channel images, poses a challenge for machine learning algorithms. Genetic programming (GP), which automates program generation, offers a potential solution to address this multimodality. However, existing GP approaches like PushGP [9, 10] and Mixed-Type Cartesian Genetic Programming (MT-CGP) [2] have limitations in handling multimodal problems effectively.

To overcome these limitations, we propose Multimodal Adaptive Graph Evolution (MAGE), an extension of MT-CGP. MAGE adopts a multi-chromosome representation, organizing function libraries specialized for distinct return types. This restructuring offers two key advantages: it reduces genome size, enhancing optimization, and accelerates convergence by aligning operands with expected function types. This structured approach improves adaptability for new data types without extensive re-engineering.

We introduce MAGE and evaluate its performance against MT-CGP on various problems from the Second Program Synthesis Benchmark Suite (PSB2) [4]. Results demonstrate that MAGE consistently converges faster and achieves superior optima compared to MT-CGP using the same set of functions. Additionally, we apply MAGE to a medical image segmentation task, showcasing its practical effectiveness.

2 BACKGROUND ON CGP AND MT-CGP

Cartesian Genetic Programming (CGP), introduced by Miller et al. [8], represents computer programs through directed acyclic graphs. The hierarchical composition of operations in CGP is achieved by decoding the network of functions mapping inputs to outputs. Similar to classical GP [5], Function libraries in CGP are tailored to specific problem requirements, such as those in computer vision for image processing and feature extraction [1, 3, 12].

In CGP, candidates are typically generated in each generation following the $1+\lambda$ Evolutionary Strategy (ES) [7]. This involves evaluating candidates with a fitness function and employing elitist selection to identify the fittest individual. The exploration of the search space primarily involves mutating individual genes within the genome of the elite. Crossover operations in CGP are not commonly employed, although some proposed solutions utilize a multi-chromosome genome [7].

CGP has traditionally addressed mono-type problems, where inputs and outputs share the same type. However, many supervised tasks involve different types, necessitating multimodal functions. Multimodal functions in CGP enable the introduction of higher-order functions, simplifying the genome. Harding et al. extended CGP to the multimodal setting with Mixed-Type Cartesian Genetic Programming (MT-CGP) [2]. MT-CGP incorporates multimodal functions alongside existing ones, streamlining the evolution process and facilitating interaction with multimodal data. It has been applied across various domains, such as sound synthesizers design and Atari game strategy learning [6, 12].

However, MT-CGP has drawbacks, including the need for functions to connect to all possible type combinations. Strategies to address this include engineering functions for all types, writing functions for each type combination, or using default values for type mismatches. These approaches make complex the construction of the function library and reduce the efficiency of mutation during evolution, prompting the introduction of Multimodal Adaptive Graph Evolution as a solution. The following section is presenting this approach.

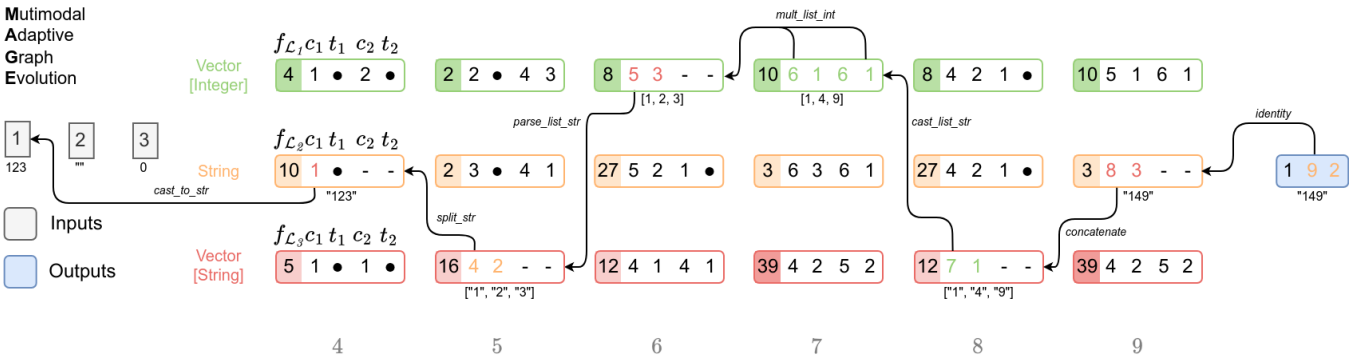

Figure 1: Encoding and graph extraction with MAGE

3 MAGE

Multimodal Adaptive Graph Evolution (MAGE) fundamentally extends the principles of CGP. Whereas MT-CGP concentrates on integrating common types, MAGE serves as a framework where all types can contribute provided the requisite type bridges are established. As MAGE represents an abstraction of CGP, it accommodates a diverse range of types across the three primary stages of a typical CGP program: inputs, operations, and outputs.

Due to the combinatorial nature of tuples (*function - operand types*) and the corresponding remedies, MT-CGP becomes less effective as the number of types or functions increases. We contend that these remedies fail to address the primary issue with MT-CGP: the use of a single vector representation for all types, which heightens the risk of encountering type mismatches at runtime. In MAGE, we have expanded the CGP representation to accommodate a broader range of types. Consequently, a MAGE genome consists of multiple chromosomes, each representing a distinct return type, mirroring the approach of PushGP in segregating stacks by types. In other words, every output originating from a node within one of the chromosomes is guaranteed to be of the same type. This assurance stems from each vector being associated with a specific function library, ensuring its corresponding return type. With access to the return type of nodes and the functions' signatures without runtime evaluation, we can guide the mutation operation and expedite the transformation from genotype to phenotype.

Consequently, a MAGE genome comprises as many chromosomes as there are function libraries. Similar to CGP, each node in the chromosome links a function to certain inputs in the graph. However, what sets MAGE apart is that inputs can explicitly originate from other types (i.e., other chromosomes), thereby fostering composition, interaction, and dynamic parameterization of functions. This seamless interconnection capability empowers MAGE to integrate multiple types at the input, program, and output levels.

The visual example presented in Figure 1 illustrates the explicit encoding approach of Multimodal Adaptive Graph Evolution (MAGE). MAGE incorporates various function libraries, each linked to a distinct chromosome, and interconnections spanning across types. The inputs resolve their concrete type at runtime, allowing for flexibility in problem-solving. The program nodes illustrate graph connections, enhancing genome decoding and evaluation speed by reducing the active graph to nodes genuinely utilized. These connections extend vertically across types and horizontally across positions. MAGE employs a "safe mutation" operator for output nodes, ensuring changes in the active node material for genuine improvements. The implementation in the Julia programming language offers a robust type system, facilitating the creation of generic functions. Additionally, MAGE introduces an "active-node-material-only" mutation, enforcing genuine changes in the active node material to exit the iterative mutation loop. Fixed output nodes are utilized to eliminate hyper-parameters and enhance results, based on recent analyses in the context of MT-CGP. Overall, MAGE's explicit encoding approach streamlines genome decoding and facilitates genuine changes in the active node material, enhancing performance in evolutionary programming tasks.

In the following section, we compare MT-CGP and MAGE on three problem taken from the PSB2 suite [4]. To our knowledge, this is the first attempt to utilize a CGP-derived method for program synthesis tasks. We observed that MAGE converges faster to a local optimum, significantly reducing the number of generations required to achieve a certain fitness score. While MT-CGP may reach a similar average fitness value for some problems, it does not converge as effectively as MAGE for others. Moreover, when both methods find absolute solutions, MAGE does so more consistently, indicating better generalization to unseen data and presenting less variance between independent runs.

4 MULTIMODAL FUNCTION SYNTHESIS

We utilized the Second Program Synthesis Benchmark (PSB2) [4] to conduct a comparative analysis between our approach and MT-CGP. We manually crafted functions that could be assembled to tackle all problems in PSB2, except for the *Bowling* problem, which we found to be excessively specific and requiring a multitude of problem-specific functions. With the knowledge that a solution exists within the realm of possible programs for each problem, we positioned MAGE and MT-CGP on equal footing and assessed the generalization potential of MAGE (i.e., how many runs were able to converge to a generalized solution). In addition to the classical libraries capable to handle integers and floating-point values, we

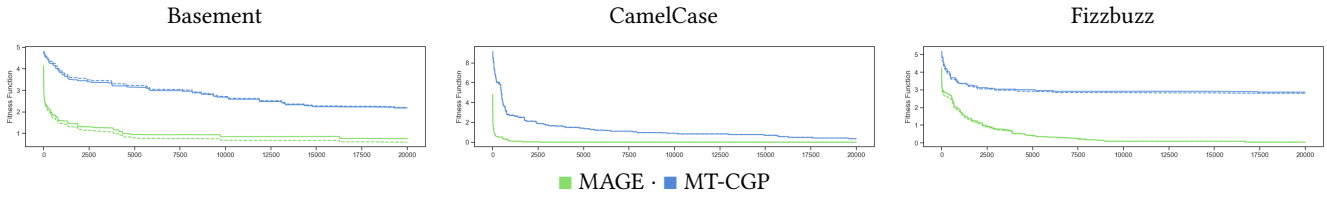

Figure 2: Comparison of the convergence of MAGE and MT-CGP on the PSB2 suite. Fitness is based on error (lower is better). Dashed lines indicate training data; solid lines indicate test data.

introduced two more complex types: *Vector{Tuple{Int,Int}}* and a corresponding one with strings. The number of functions in the *Float* and *Integer* libraries is identical because they share the same function set. The same principle applies to the *Vector{Tuple{* $\cdot, \cdot$ *}}* libraries and the majority of functions dealing with vectors.

In this paper, we compared MT-CGP and MAGE across 3 problems (Basement, CamelCase and Fizzbuzz) in the PSB2 benchmark utilizing the fitness functions described in [4]. Each algorithm underwent evaluation through 20 independent runs, totaling 400 runs. Shared parameters were maintained constant for both MAGE and MT-CGP. To minimize variability in mutation probabilities, we introduced a *real-numbered* mutation operation, which mutates precisely that number of alleles in the active graph. If the active graph contains fewer active nodes than the specified mutation parameter, we mutate all active nodes. For all experiments, a mutation parameter of 1 was used, indicating that only one allele in the active graph is mutated at each mutation step.

Notably, three points distinguish MT-CGP from MAGE. In MAGE, we employ a *safe mutation* operation, whereas in MT-CGP, nodes are mutated without restrictions. Mutations are performed solely on the active part of the node in MAGE, ensuring that modifications in the encoding representation result in genuine changes. Secondly, prior to the first iteration, all nodes in the initial MAGE genome are corrected (i.e., mutated until they are repaired). Thirdly, the genome size for MAGE was fixed at 30 for every problem. However, a MAGE genome of size 30 with 2 function libraries comprises a total of 60 nodes. To maintain consistency, MT-CGP is allocated 30 nodes per type to match the total number of nodes present in the MAGE configuration.

In Figure 2, and table 1, we compare the fitness over training on three different problems (Basement, CamelCase and Fizzbuzz) taken from the PSB-2 benchmark suite, comparing MT-CGP and MAGE. MAGE outperforms MT-CGP on all three metrics for the three presented problems. MAGE also converges much more rapidly than MT-CGP, requiring thousands fewer iterations to reach a certain score. Each problem problem has been tested 20 times to evaluate the robustness of our approach.

	Solved		Generation	
Problem	**MAGE**	**MT-CGP**	**MAGE**	**MT-CGP**
Basement	12	1	4278	18339
CamelCase	20	15	461	10514
FizzBuzz	17	0	6763	-

Table 1: Number of runs which fully solved the given three PSB2 problems, out of 20. The average generation only concerns the successful runs.

These differences between MAGE and MT-CGP stem from the more directed and assisted search in MAGE. With the multiplication of possible combinations between inputs of different types and functions involved in the usage of multiple function libraries, the search space of MT-CGP becomes challenging to navigate. In MT-CGP, there is always a high likelihood that mutation introduces a discrepancy between function and input types, which can break or hinder its final output. Conversely, MAGE only produces individuals with consistent types, allowing evolution to proceed to relevant solutions rapidly.

5 MEDICAL IMAGE SEGMENTATION

Many computer vision applications can be considered multimodal because the interaction between arrays, vectors, kernels, and scalar pixel values can be crucial for discovering simple, generalizable, and explainable pipelines. Therefore, we assess the ability of MAGE to perform image segmentation on biomedical images.

For this purpose, we integrate MAGE into the Kartezio framework [1], an image processing library based on CGP. In Kartezio, CGP utilizes an image processing function library composed of 43 functions, including image filters and transformations. The MAGE model incorporates two additional function libraries: $\mathcal{L}_a$, with 44 functions, returning arrays, and $\mathcal{L}_s$, with 17 functions, returning scalars. $\mathcal{L}_a$ is based on the original Kartezio image processing library, but it replaces most parameters with a scalar input. Additionally, the MAGE model includes two default scalar inputs: 0.0 and 1.0. Both models take two channels as inputs, Phalloidin and DAPI, and utilize a Watershed transform on the program output. This transformation requires two inputs: the markers and the topological mask, to produce the final instance segmentation prediction as described in [1].

To evaluate MAGE, we utilized the P2043 Cell Image Library dataset, generously provided by the Cellpose authors [11]. This dataset comprises 89 training images and 11 test images.To compare the two models, we randomly selected 8 images from the training set (with the same sampling for both models) and conducted 10 independent runs. The fitness function employed is a composite of Average Precision (with a threshold of 0.5) (AP50) and Intersection over Union (IoU) to smooth the fitness landscape: $fitness = 1.0 \times \text{AP50} + 0.1 \times \text{IoU}$.

While the mean AP50 value on the training set are equivalent (0.8905 for MAGE and 0.8934 for CGP), MAGE outperforms CGP

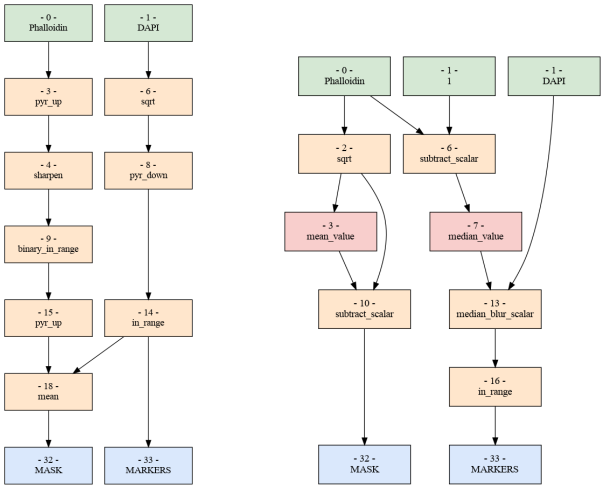

Figure 3: An example of evolved mono-typed CGP graph (left) and MAGE (right) for Instance Segmentation

across the test data with a mean AP50 value of 0.8264 for MAGE against 0.8161, demonstrating better generalization capabilities. Additionally, in Figure 3, we illustrate the evolved graph for CGP (left) and the one for MAGE (right). Although the utilization of scalar parameters is minimal, involving only two scalar functions, we posit that these dynamic parameters contribute to the observed improvement in generalization and the ability to discern nearby cells.

6 CONCLUSION

Multimodal problems are prevalent across various real-world applications, presenting challenges for Cartesian Genetic Programming (CGP) due to its difficulty in handling diverse data types. Mixed-Type Cartesian Genetic Programming (MT-CGP) emerged as a solution, adapting CGP for multimodal settings, especially in vision applications. However, MT-CGP encountered limitations as problems became more complex, such as struggles to define functions accommodating multiple types and susceptibility to getting trapped in suboptimal solutions due to an expanding search space. Moreover, reliance on problem-specific solutions hindered collaboration in the field.

In response, we introduced Multimodal Adaptive Graph Evolution (MAGE), a generalized CGP approach that segregates genome chromosomes by types, akin to how PushGP treats stacks. This separation optimizes convergence by enabling the mutation operation to assess node states more effectively and enhances individual performance through pre-fixing before evaluation.

Through a comparative study between MT-CGP and MAGE in multimodal program synthesis tasks using the PSB2 benchmark, we found that MAGE consistently outperformed or matched MT-CGP across various metrics. It demonstrated superior generalization capabilities and faster convergence, regardless of MT-CGP's performance. Overall, MAGE represents a significant advancement in GP evolution by efficiently addressing multimodal problems, offering a robust approach applicable across different domains. Specifically, in biomedical data analysis, MAGE's ability to handle diverse data types and generate interpretable graphs holds promise for providing verifiable solutions crucial for both medical professionals and industry stakeholders.

Acknowledgments

This project used compute resources from the CALMIP project P21049.

REFERENCES

[1] Kévin Cortacero, Brienne McKenzie, Sabina Müller, Roxana Khazen, Fanny Lafouresse, Gaëlle Corsaut, Nathalie Van Acker, François-Xavier Frenois, Laurence Lamant, Nicolas Meyer, Béatrice Vergier, Dennis G. Wilson, Hervé Luga, Oskar Staufer, Michael L. Dustin, Salvatore Valitutti, and Sylvain Cussat-Blanc. 2023. Evolutionary design of explainable algorithms for biomedical image segmentation. *Nature Communications* 14, 1 (Nov. 2023), 7112. https://www.nature.com/articles/s41467-023-42664-x Number: 1 Publisher: Nature Publishing Group.

[2] Simon Harding, Vincent Graziano, Jürgen Leitner, and Jürgen Schmidhuber. 2012. MT-CGP: mixed type cartesian genetic programming. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation (GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 751–758.

[3] Simon Harding, Jürgen Leitner, and Jürgen Schmidhuber. 2013. Cartesian Genetic Programming for Image Processing. In *Genetic Programming Theory and Practice X*, Rick Riolo, Ekaterina Vladislavleva, Marylyn D Ritchie, and Jason H. Moore (Eds.). Springer, New York, NY.

[4] Thomas Helmuth and Peter Kelly. 2021. PSB2: the second program synthesis benchmark suite. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lille, France) *(GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 785–794.

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

[6] Matthieu Macret and Philippe Pasquier. 2014. Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14)*. Association for Computing Machinery, New York, NY, USA, 309–316.

[7] Julian Francis Miller. 2020. Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines* 21, 1 (June 2020), 129–168.

[8] Julian F. Miller and Peter Thomson. 2000. Cartesian Genetic Programming. In *Genetic Programming (Lecture Notes in Computer Science)*, Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian Miller, Peter Nordin, and Terence C. Fogarty (Eds.). Springer, Berlin, Heidelberg.

[9] Lee Spector, Jon Klein, and Maarten Keijzer. 2005. The Push3 execution stack and the evolution of control. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (Washington DC, USA) *(GECCO '05)*. Association for Computing Machinery, New York, NY, USA, 1689–1696.

[10] Lee Spector and Alan Robinson. 2002. Genetic Programming and Autoconstructive Evolution with the Push Programming Language. *Genetic Programming and Evolvable Machines* 3, 1 (March 2002), 7–40.

[11] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. 2021. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods* 18, 1 (2021), 100–106.

[12] Dennis G Wilson, Sylvain Cussat-Blanc, Hervé Luga, and Julian F Miller. 2018. Evolving simple programs for playing atari games. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. Association for Computing Machinery, New York, NY, USA.