TUTORIAL

Cartesian Genetic Programming: From foundations to recent developments and applications

ROMAN KALKREUTH, RWTH Aachen University, Aachen, Nordrhein-Westfalen, Germany

SYLVAIN CUSSAT-BLANC, Toulouse Capitole University, Toulouse, Occitanie, France

DENNIS G WILSON, University of Toulouse, Toulouse, Occitanie, France

Published: 14 July 2025

Citation in BibTeX format

GECCO '25 Companion: Genetic and Evolutionary Computation Conference Companion
July 14 - 18, 2025
Malaga, Spain

Conference Sponsors:
SIGEVO

https://doi.org/10.1145/3712255.3716540
ISBN: 9798400714641

Cartesian Genetic Programming: From Foundations to Recent Developments and Applications

Roman Kalkreuth, Sylvain Cussat-Blanc, Dennis G. Wilson

RWTH Aachen University, Aachen, Germany

University of Toulouse, Toulouse France

ISAE-Supaero, Toulouse, France

roman.kalkreuth@rwth-aachen.de
sylvain.cussat-blanc@irit.fr
dennis.wilson@isae.fr

ACM ISBN 979-8-4007-1464-1/2025/07
doi:10.1145/3712255.3716540

About us

Roman Kalkreuth is an assistant professor at RWTH Aachen University (Germany). Primarily, his research focuses on the analysis and development of algorithms for graph-based genetic programming.

Sylvain Cussat-Blanc is a professor at University Toulouse Capitole, France. He has a junior chair at the Institut Universitaire de France in the use of Cartesian Genetic Programming applied to biomedical data analysis, and in particular biomedical images.

Dennis Wilson is an associate professor at ISAE-Supaero in Toulouse, France. His research focuses on learning and optimization in artificial systems, with applications in climate change.

Agenda

- Origin and context of graph-based GP
- Overview of CGP
- Practical demonstration in Python[1]
- Break
- Applications of CGP
- Status and Future of CGP
- Questions and Discussion

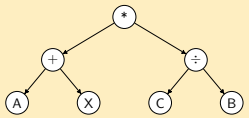
[1] https://github.com/d9w/CGP-tutorial

Objectives of this Tutorial

- **Provide a comprehensive introduction to CGP**
  - Historical context and evolution from other GP methods
  - Understanding CGP's position within evolutionary computation
  - Key advantages over other representation schemes
- **Explore the fundamental mechanisms of CGP**
  - Graph-based representation and genotype-phenotype mapping
  - Mutation operators and their effect on search dynamics
  - The role of neutrality and inactive genes
  - Advances in recombination techniques
- **Demonstrate CGP's versatility across applications**
  - Digital circuit design and optimization
  - Image processing and computer vision
  - Machine learning model generation
  - Neuroevolution and biological modeling
- **Provide practical implementation guidance**
  - Parameter selection and tuning strategies
  - Available software tools and resources
  - Common pitfalls and how to avoid them
  - Future research directions and open questions

Genetic Programming (GP)

- Genetic Programming is a search heuristic
- Inspired by neo-Darwinian evolution
- Method for the synthesis of computer programs
- Traditionally used with parse trees



$\mathcal{F} = \{ +, -, *, \div \}$
$\mathcal{T} = \{ A, X, C, B \}$
$\mathcal{E} = \text{Edges}$
$\Psi = (A + X) * (C \div B)$

Definition (Genetic Programming)

Let Θ be a population of $|\Theta|$ individuals and let Ω be the population of the following generation:

- Each individual is represented with a **genetic program** and a **fitness value**
- Genetic Programming transforms $\Theta \mapsto \Omega$ by the adaptation of **selection**, **recombination** and **mutation**

Definition (Genetic Program)

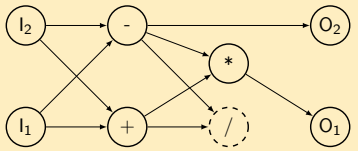
A genetic programm $\mathcal{P}$ is an element of $\mathcal{T} \times \mathcal{F} \times \mathcal{E}$:

- $\mathcal{F}$ is a finite non-empty set of functions
- $\mathcal{T}$ is a finite non-empty set of terminals
- $\mathcal{E}$ is a finite non-empty set of edges

Let $\phi : \mathcal{P} \mapsto \Psi$ be a decode function which maps $\mathcal{P}$ to a phenotype Ψ

What we mean by graph-based

- Genetic Programming $\rightarrow$ traditionally tree representation, which is a **well defined form of graph**
- Graph-based Genetic Programming $\rightarrow$ use of graph representation models that **extend GP beyond trees**
- In the methods we consider, **Directed Acyclic Graphs** (DAGs), introducing:
  - Reuse of intermediate results.
  - Active and inactive nodes.



Introduction and Related Work
Historical Background

Graph-based GP Timeline

1990	Koza: Genetic Programming on LISP Syntax Trees
1993	Banzhaf: Precursor to Linear Genetic Programming
1994	Nordin: Compiling Genetic Programming System
1996	Poli: Parallel Distributed Genetic Programming
1996	Teller: Parallel Algorithm Discovery and Orchestration (PADO)
1997	Miller, Thompson, Kalganova, Fogarty: Steps forward Cartesian Genetic Programming
1998	Nordin: Automatic Induction of Machine Code with Genetic Programming
1999	Angeline: Multiple Interacting Programs (MIPs)
1999	Miller. Thompson: **Cartesian Genetic Programming**
2001	Brameier, Banzhaf: **Linear Genetic Programming**
2002	Kantschik, Banzhaf: Linear-Graph Genetic Programming
2008	Galvan-Lopez: Multiple Interactive Outputs in a Single Tree (MIOST)
2011	Downey, Zhang: Parallel Linear Genetic Programming
2017	Kelly and Keywood: Tangled Program Graphs (TGP)
2018	Atkinson, Plump, Stepney: **Evolving Graphs by Graph Programming**

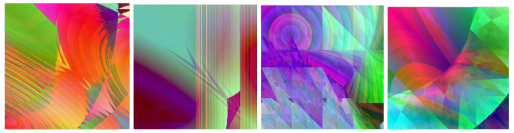
Cartesian Genetic Programming
Historical Background

Cartesian GP Timeline

1997	Miller, Thompson, Kalganova, Fogarty: Work towards CGP
1999	Miller, Thompson: **Standard CGP**
2004	Walker, Miller: Modular CGP
2005	Smith, Leggett, Tyrrell: Implicit Context Representation of CGP
2007	Clegg, Walker, Miller: Real-valued representation of CGP
2008	Walker, Miller: Embedded CGP
2011	Harding, Miller, Banzhaf: Self-Modifying CGP
2011	Harding, Graziano, Leitner, Schmidhuber: Multi-type CGP
2013	Turner, Miller: CGP encoded artificial neural networks
2014	Turner, Miller: Recurrent CGP
2016	Ryser-Welch, Miller, Swan and Trefzer: Iterative CGP
2018	Wilson, Miller, Cussat-Blanc, Luga: Positional CGP

Cartesian Genetic Programming (CGP)
Historical Background

- First work towards CGP was done by Miller, Thompson, Kalganova, and Fogarty [21, 43, 44]
- Represents genetic programs as a two dimensional grid of program primitives
- Loosely inspired by the architecture of digital circuits



Automatically evolved images with CGP (Ashmore and Miller [6])

Cartesian Genetic Programming (CGP)
Representation Model

- Program representation $\rightarrow$ acyclic and directed graph.
- Genotype-phenotype mapping $\rightarrow$ encoding-decoding of the graph
- Predominantly used without recombination $\rightarrow$ mutational $(1+\lambda)$ evolutionary algorithm.

Cartesian Genetic Programming (CGP)
Representation Model

Definition (Cartesian Genetic Program (CP))

A cartesian genetic program $\mathcal{P}$ is an element of $\mathcal{N}_i \times \mathcal{N}_f \times \mathcal{N}_o \times \mathcal{F}$:

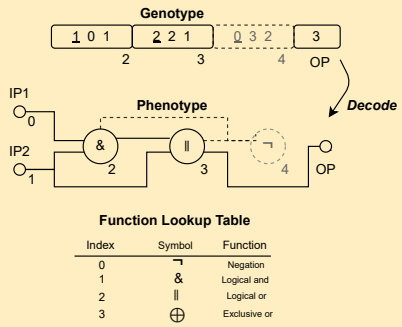
- $\mathcal{N}_i$ is a finite non-empty set of input nodes
- $\mathcal{N}_f$ is a finite set of function nodes
- $\mathcal{N}_o$ is a finite non-empty set of output nodes
- $\mathcal{F}$ is a finite non-empty set of functions

Cartesian Genetic Programming (CGP)
Representation Model

- Nodes of a cartesian genetic program are **continuously indexed**
- Indexing starts with the a value of 0 at the **first input node** and ends at the **last output node**
- At **each node**, the node number is **increased by one**
- Let $N = |N_i| + |N_f| + |N_o|$ be the number of nodes of a CP

Cartesian Genetic Programming
Representation Model

Cartesian Genetic Programming



Function Lookup Table

Index	Symbol	Function
0	¬	Negation
1	&	Logical and
2	‖	Logical or
3	⊕	Exclusive or

Cartesian Genetic Programming (CGP)
Search Algorithm

- CGP is commonly used with a variant of the $(1 + \lambda)$-EA $\rightarrow$ $(1+\lambda)$-CGP
- Implements a modified selection strategy called **neutrality**
- Adapts a **genetic drift** to provide diversity during the evolutionary run
- Individuals that have the same fitness are determined, and **one** of these **same-fitness individuals** is returned **uniformly at random**

Cartesian Genetic Programming (CGP)
Search Algorithm

Algorithm 1 (1+λ)-EA variant used in CGP

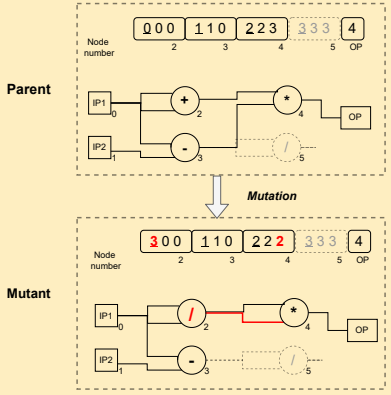
1: initialize($\mathcal{P}$) ▷ *Initialize parent individual*
2: **repeat** ▷ *Until termination criteria not triggered*
3: $\mathcal{Q} \leftarrow breed(\mathcal{P})$ ▷ *Breed λ offspring by mutation*
4: Evaluate($\mathcal{Q}$) ▷ *Evaluate the fitness of the offspring*
5: $\mathcal{Q}^+ \leftarrow best(\mathcal{Q}, \mathcal{P})$ ▷ *Get individuals which have better fitness as the parent*
6: $\mathcal{Q}^= \leftarrow equal(\mathcal{Q}, \mathcal{P})$ ▷ *Get individuals which have the same fitness as the parent*
7: ▷ *If there exist individuals with better fitness*
8: **if** $|\mathcal{Q}^+| > 0$ **then**
9: ▷ *Choose one individual from $\mathcal{Q}^+$ uniformly at random*
10: $\mathcal{P} \leftarrow \mathcal{Q}^+[r]$, $r \sim U[0, |\mathcal{Q}^+| - 1]$
11: ▷ *Otherwise, if there exist individuals with equal fitness*
12: **else if** $|\mathcal{Q}^=| > 0$ **then**
13: ▷ *Choose one individual from $\mathcal{Q}^=$ uniformly at random*
14: $\mathcal{P} \leftarrow \mathcal{Q}^=[r]$, $r \sim U[0, |\mathcal{Q}^=| - 1]$
15: **end if**
16: **until** $\mathcal{P}$ *meets termination criterion*
17: **return** $\mathcal{P}$ ▷

Cartesian Genetic Programming (CGP)
Mutation

- Standard genetic operator → probabilistic **point mutation**
- Genes are **selected uniformly at random** in the genotype
- **Exchange of gene values** in the valid range by chance
- Genetic **variation** of **functionality** and **connectivity**

Cartesian Genetic Programming (CGP)
Mutation

Example of standard point mutation in CGP



Cartesian Genetic Programming (CGP)
Mutation

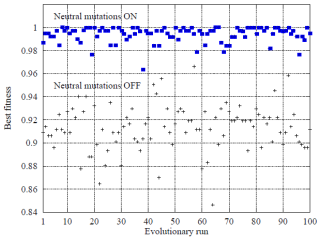
Algorithm 2 Probabilistic point mutation

Input: Genome $\mathcal{G}$, Function set $\mathcal{F}$, Number of function nodes $\mathcal{N}_f$, Number of input nodes $\mathcal{N}_i$, Mutation rate $\mathcal{P}$
Output: Mutated Genome $\widetilde{\mathcal{G}}$

1: **foreach** $g \in \mathcal{G}$ **do** ▷ *Iterate over the genome*
2: $X \sim \text{Ber}(\mathcal{P})$ ▷ *Bernoulli random variable*
3: **if** $X = 1$ **then** ▷ *Control the mutation strength with X*
4: **if** g is a connection gene **then**
5: ▷ *Determine the node number of the gene*
6: $n \leftarrow \text{NodeNumber}(g)$
7: ▷ *Select g in the range of previous node indexes by chance*
8: $g \leftarrow r$, $r \sim U[0, n - 1]$
9: **else if** g is a function gene **then**
10: ▷ *Select g in the range of the function indexes by chance*
11: $g \leftarrow r$, $r \sim U[0, |F| - 1]$
12: **else** ▷ *g is a output gene*
13: ▷ *Select g in the range of the function and input nodes by chance*
14: $g \leftarrow r$, $r \sim U[0, |N_f| + |N_i| - 1]$
15: **end if**
16: **end if**
17: **end foreach**
18: **return** $\mathcal{G}$ ▷ *Return the mutated genome*

Cartesian Genetic Programming (CGP)
Mutation

- A number of studies [56, 55, 45] have demonstrated the importance of **neutral mutations** that are based on **functional redundancy**
- Functional redundancy refers to many different programs representing the same function



Evolution of three-bit Boolean multiplier: Obtained fitness is show for each of 100 runs of 10 million generations with neutral mutations enabled (blue) compared with disabled neutral mutations (+)

Cartesian Genetic Programming (CGP)
Mutation

- **Single active gene mutation (SAM)** is a variant of standard point mutation introduced by Goldman and Punch [17].
- SAM mutates the genome until an active gene is hit
- Major advantage → **no parametrization** of the mutation rate is **needed**
- Let $g_a = n_a * (1 + a) + n_o$ be the number of active genes, the active gene which will be mutated is selected uniformly at random with probability $\frac{1}{g_a}$
  - n_a is the number of active nodes.
  - n_o is the number of outputs.
  - a is the maximum arity of a function node.

Cartesian Genetic Programming (CGP)
Mutation

Algorithm 3 Single active gene mutation

Input: Genome $\mathcal{G}$, Function set $\mathcal{F}$, Number of function nodes $\mathcal{N}_f$, Number of input nodes $\mathcal{N}_i$
Output: Mutated Genome $\widetilde{\mathcal{G}}$

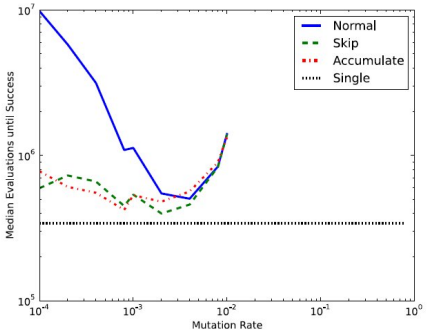
active ← **false**
repeat
  $g \leftarrow \mathcal{G}[r]$, $r \sim U[0, |\mathcal{G}| - 1]$ ▷ *Select a gene by chance*
  $n \leftarrow$ NodeNumber(g) ▷ *Determine the node number of the gene*
  active ← NodeActive(n) ▷ *Check whether gene is active or not*
  if g is a connection gene **then**
    ▷ *Select g in the range of previous node indexes by chance*
    $g \leftarrow r$, $r \sim U[0, n - 1]$
  else if g is a function gene **then**
    ▷ *Select g in the range of the function indexes by chance*
    $g \leftarrow r$, $r \sim U[0, |F| - 1]$
  else ▷ *g is a output gene*
    ▷ *Select g in the range of the function and input nodes by chance*
    $g \leftarrow r$, $r \sim U[0, |N_f| + |N_i| - 1]$
  end if
until active is **true**

Cartesian Genetic Programming (CGP)
Mutation

- Goldman and Punch [17] compared several mutation strategies on a set of Boolean functions → Standard, Skip, Accumulate and SAM
- **Standard** is the standard (probabilistic) point mutation
- **Skip** sets the offspring fitness to parent's fitness if phenotypes are identical
- **Accumulate** mutates multiple genes until some active genes are changed

Cartesian Genetic Programming (CGP)
Mutation

- SAM was close to the best performance of the other strategies on the majority of the tested problems
- Showed good performance on the 3 bit parallel multiplier problem

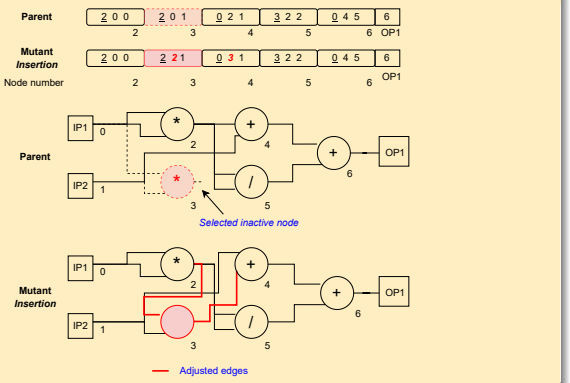


Cartesian Genetic Programming (CGP)
Mutation

- Kalkreuth [22, 24, 25, 27] adapted a set of chromosomal mutations: **insertion**, **deletion**, **inversion** and **duplication**
- **Insertion** selects an **inactive** node and changes one or more connection genes in the genotype to make it **active**
- **Deletion** alters connections to an active node so that the node becomes inactive
- **Inversion** permutates the function gene values of a set of successive active function nodes
- **Duplication** adapts chromosomal duplication by replacing active function genes

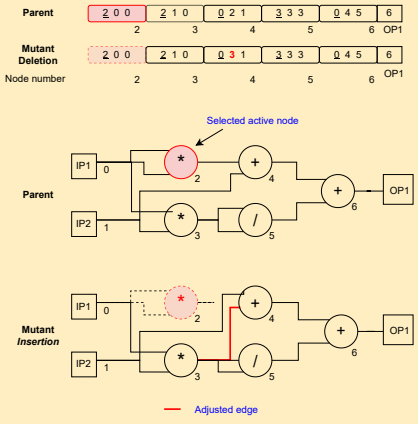
Cartesian Genetic Programming (CGP)
Mutation

Insertion Mutation



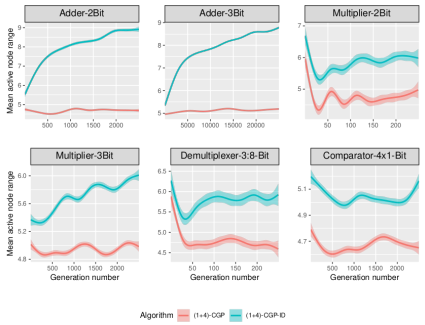
Cartesian Genetic Programming (CGP)
Mutation

Deletion Mutation



Cartesian Genetic Programming (CGP)
Mutation

- Evaluation on a diverse set of Boolean function problems
- Search performance improved significantly on the majority of the tested problems
- Insertion and deletion **increase** the mean **active node range** of the evolutionary search



Cartesian Genetic Programming (CGP)
Recombination or not? A long and ongoing story...

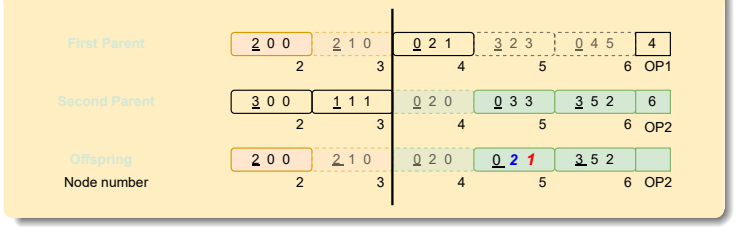
- Miller [44] investigated **genotypic recombination** but observed that **it does not seem to add anything**
- Recombination on **multiple chromosomes** with **independent fitness assessment**has been found **highly beneficial**
- Clegg et al. [10] proposed a **floating point representation** of CGP and suggested that **intermediate recombination** might be useful for **symbolic regression**
- **Cone-based crossover** has been proposed for modular CGP by Kaufmann et al. [33]
- da Silva et al. [13] proposed **path recombination** for Boolean **multiple-output problems** and obtained better search performance on the **single-chromosome representation**
- Kalkreuth [30, 23] found that **subgraph** [28] and **function gene recombination** [20] can improve the search performance for various **symbolic regression problems**

Cartesian Genetic Programming (CGP)
Recombination or not? A long and ongoing story...

- Kalkreuth [26] proposed **discrete recombination**
  - Improved search performance was obtained on a set of symbolic regression problems
- Cui [12] **compared** the use of **various crossover operators** to **mutation-only** CGP
  - Better performance metrics were obtained for CGP **with crossover**
    - **No universally best-performing** crossover operator
  - Hyperparameter optimization is essential
  - CGP with recombination might not negatively impact its evolutionary search process
  - When CGP (with or without recombination) **uses tournament selection** on logic synthesis benchmarks, the **performance strongly degrades**

Cartesian Genetic Programming (CGP)
Recombination or not? A long and ongoing journey...

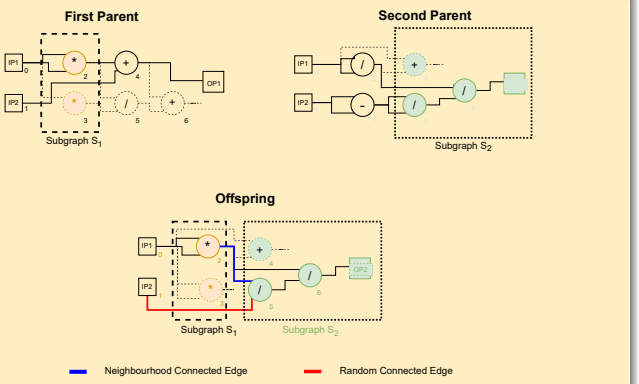
Subgraph Crossover



Cartesian Genetic Programming (CGP)

Recombination or not? A long and ongoing story...

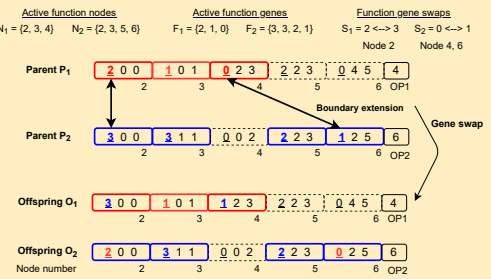
Subgraph Crossover



Cartesian Genetic Programming (CGP)

Recombination or not? A long and ongoing journey...

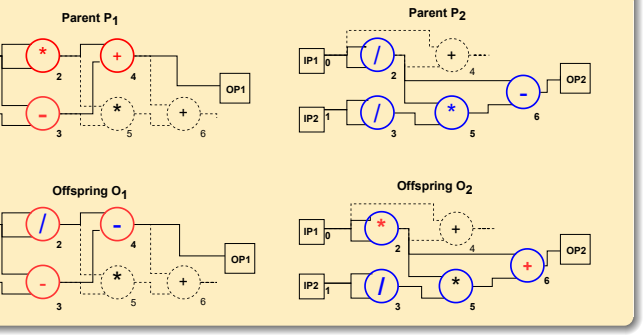
Discrete Recombination



Cartesian Genetic Programming (CGP)

Recombination or not? A long and ongoing story...

Discrete Recombination



Cartesian Genetic Programming (CGP)

Parametrization: Genotype Length

- The effectiveness of the search in relation to the number of nodes and mutation rate was investigated on a couple of **Boolean functions** [45].
- **Efficiency** of the search appeared to **continuously improve** as the **number of nodes increased**
  - Least number of evaluations required for success when 95 % of the genes were inactive
- However, Kalkreuth [30] demonstrated that small genotypes can perform effectively in the **symbolic regression** domain

Cartesian Genetic Programming (CGP)
Parametrization: Population size

- Various hyperparameter settings have been extensively investigated [31, 32, 30]
- Evaluating Boolean functions and symbolic regression problems it was found:
  - $\lambda = 1$ was often the most effective setting for the tested Boolean function problems
  - Higher settings (greater than 4) of λ performed effective on various symbolic regression problems

Cartesian Genetic Programming (CGP)
Parametrization: Benchmarking CGP

- The *General Boolean Function Benchmark Suite* (GBFS) has been proposed as a benchmark for (C)GP in logic synthesis [29]
  - Covers different types of Boolean functions commonly used in the field of GP
  - Thoughtfully selected Boolean functions from seven popular categories
  - GBFS provides **38 benchmark functions** in total[2]

Identifier	Name	Category
add	Adder with carry	Arithmetic
mul	Multiplier	Arithmetic
dec	One-hot decoder	Transmission
enc	One-hot encoder	Transmission
icomp	Identity Comparator	Comparison
mcomp	Magnitude Comparator	Comparison
count	Ones' counter circuit	Counting
alu	Arithmetic Logic Unit	Mixed
epar	Parity Even	Parity
bent	Bent	Cryptography
bal	Balanced	Cryptography
res	Resilient	Cryptography
mask	Masking	Cryptography

[2] https://github.com/boolean-function-benchmarks

Cartesian Genetic Programming (CGP)
Parametrization: Benchmarking CGP

- Confirmed previous findings related to CGP's efficiency on a larger benchmark
  - Merely focused on mutation-only CGP
- HPO experiments indicate that setting $\lambda = 1$ is a general effective choice
- Number of function nodes (genotype length) appears to be strongly problem dependent
  - Large genotypes performed best
- Performing HPO for each problem is highly recommended to ensure fair conditions in comparative studies

Cartesian Genetic Programming (CGP)
Example in Python

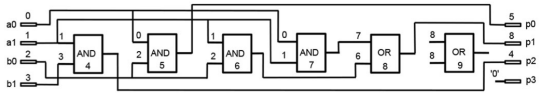
https://github.com/d9w/CGP-tutorial

Applications of CGP
Outline

- Digital circuit synthesis
- Machine learning
- Neural architecture search
- Image processing

Digital circuit synthesis

- Original application of CGP [46]
- Widely used in the literature as benchmark [53, 18, 54, 34, 49, 19]
- Function library composed of logic gates (AND, OR, NOT)
- The graph represents the logic circuit
- Potential room for multi-objective optimization [8]:
  - Correctness: matching against a given truth table
  - Cost: Number of gates used or power consumption
- Recent extension to quantum logic [15]
- Multiple prizes won at the Humies Awards
  - Silver Medal at GECCO 2011 [53]
  - Bronze Medal at GECCO 2014 [18]
  - Gold Medal at GECCO 2015[54]

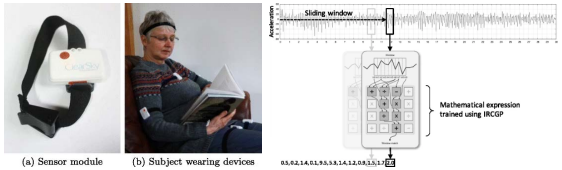


Machine learning
Regression - Classification

- Symbolic regression is a straight forward application of CGP
- Function library usually composed of basic mathematical function:
  - $+$, $-$, $*$, $/$,
  - *sin*, *cos*, *tan*,
  - *log*, *exp*, *sqrt*, *pow*,
  - *min*, *max*.
- Fitness function usually based on MSE or MAE

Machine learning
Regression - Classification

- Many scientific and industrial applications:
  - System control [7]
  - Space [42]
  - Physical model discovery and improvement [14]
- Humies Gold Medal Award at GECCO 2018 when applied to Parkinson's Dyskinesia classification[40]



(a) Sensor module (b) Subject wearing devices

Machine learning

Feature selection & construction

- CGP has been successful for automated feature engineering
- Functions effectively as preprocessing step before traditional ML algorithms
- Advantages over manual feature engineering:
  - Automatic exploration of large feature space
  - Discovery of non-intuitive feature combinations
  - Reduction of dimensionality while preserving information
- Applications:
  - Biomarker identification in medical data
  - Signal processing and time-series analysis
  - Classification with interpretable features
  - Noise filtering and preprocessing
- Search space includes mathematical operations, filters, and transformations applied to raw input features

Applications in Evolutionary Robotics

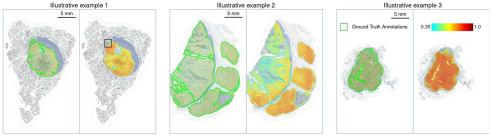
- Control of legged robots and autonomous vehicles
  - Evolving gait controllers for quadruped and hexapod robots
  - Navigation in complex environments without precise world models
  - Adaptation to hardware failures (e.g., broken legs or sensor malfunctions)
- Swarm robotics and multi-agent coordination [9]
  - Distributed control with limited communication
  - Self-organization and emergent behaviors
  - Task allocation and resource management
- Advantages over traditional control approaches:
  - No need for analytical models of the robot or environment
  - Robust to noise and uncertainty
  - Can discover novel control strategies beyond human design
  - Adaptable to changing environments and robot configurations

Neural architecture search - CGPANN

- Typically used to produce CNN architectures [37, 36, 51, 35]
- Neural architecture encoded inside a CGP genome [52, 50, 41, 38]
  - Each node represents a layer (conv, pooling, dense, etc.)
  - Connection between (active) nodes represent connection between layers
- Potential room for multi-objective optimization [16]
- Evolution of neural development rules [47]
- Many fields of applications:
  - Financial [57]
  - Medical [2, 1, 3, 4]
  - Client prediction [5]
  - Networks [48]
  - etc.

Image processing

- Segmentation or instance segmentation of images [**?**, 39, 11]
- Function library composed of:
  - standard functions (*min*, *max*, *mean*, *add*, etc.)
  - mask processing functions (*bitwise_*{*not*, *or*, *and*, *mask*}, *threshold*, etc.)
  - image processing functions (*sobel*, *canny*, *gabor*, etc.)
  - morphologic functions (*morph_*{*gradient*, *tophat*, *blackhat*}, *erode*, *dilate*, *fill_hole*, etc.)
- Fitness functions based on IOU or AP
- Requires very small amount of annotated images ($\tilde{10}$) for optimization
- Humies Gold Medal Award at GECCO 2024[11]



Questions and Discussion

Open research questions in CGP

- **Mutation strategies:**
  - What is the optimal balance between exploration and exploitation?
  - When should we use single active gene mutation vs. point mutation?
  - Are there domain-specific mutation operators that could improve performance?
- **Recombination:**
  - When is crossover beneficial in CGP and when is it detrimental?
  - What are the most effective crossover methods for specific domains?
  - How can we design recombination operators that respect the graph structure?
- **Search algorithms:**
  - Beyond $(1 + \lambda)$-ES, what search strategies are promising?
  - Is neutrality always beneficial or are there cases where it hinders search?
  - How to balance between genotype size and search efficiency?

Questions and Discussion

Challenges and future directions

- **Scalability issues:**
  - How to evolve larger programs efficiently?
  - Techniques for handling high-dimensional input/output spaces?
  - Parallelization and distributed evolution strategies?
- **Interpretability:**
  - Tools for analyzing evolved CGP programs
  - Automatic simplification of evolved solutions
  - Visualization techniques for complex graphs
- **Hybrid approaches:**
  - Combining CGP with deep learning
  - Using CGP as a meta-optimizer for other ML methods
  - Multi-objective optimization in CGP

Questions and Discussion

Research opportunities and applications

- **Hardware implementation:**
  - Designing energy-efficient circuits for IoT and embedded systems
  - Evolvable hardware for adaptive systems
  - Specialized hardware accelerators for CGP
- **Explainable AI:**
  - CGP as an alternative to black-box deep learning
  - Generating human-readable decision processes
  - Application in regulated domains (healthcare, finance, etc.)

Questions and Discussion

Getting started with CGP

- **Available software and libraries:**
  - CGP Library (in C) by Andrew Turner
  - CGP for Python, Julia, MATLAB implementations
  - Domain-specific implementations (e.g., for neural networks)
- **Recommended benchmarks:**
  - Boolean function synthesis (even-parity, multiplier)
  - Symbolic regression problems
  - Classification tasks
  - Control problems
- **Tips for practical implementation:**
  - Start with small genotypes and gradually increase size
  - Carefully select function set for your domain
  - Monitor neutral drift during evolution
  - Consider computational costs in fitness evaluation

References I

Arbab Masood Ahmad and Gul Muhammad Khan.
Bio-signal processing using cartesian genetic programming evolved artificial neural network (cgpann).
In *2012 10th International Conference on Frontiers of Information Technology*, pages 261–268. IEEE, 2012.

Arbab Masood Ahmad, Gul Muhammad Khan, Sahibzada Ali Mahmud, and Julian Francis Miller.
Breast cancer detection using cartesian genetic programming evolved artificial neural networks.
In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 1031–1038, 2012.

Arbab Masood Ahmad, Gul Muhammad Khan, and Sahibzada Ali Mahmud.
Classification of arrhythmia types using cartesian genetic programming evolved artificial neural networks.
In *Engineering Applications of Neural Networks: 14th International Conference, EANN 2013, Halkidiki, Greece, September 13-16, 2013 Proceedings, Part I 14*, pages 282–291. Springer, 2013.

Arbab Masood Ahmad, Gul Muhammad Khan, and Sahibzada Ali Mahmud.
Classification of mammograms using cartesian genetic programming evolved artificial neural networks.
In *Artificial Intelligence Applications and Innovations: 10th IFIP WG 12.5 International Conference, AIAI 2014, Rhodes, Greece, September 19-21, 2014. Proceedings 10*, pages 203–213. Springer, 2014.

Jawad Ali, Faheem Zafari, Gul Muhammad Khan, and S Ali Mahmud.
Future clients' requests estimation for dynamic resource allocation in cloud data center using cgpann.
In *2013 12th International Conference on Machine Learning and Applications*, volume 2, pages 331–334. IEEE, 2013.

Laurence Ashmore and J Miller.
Evolutionary art with cartesian genetic programming.
Technical Online Report, 2004.

References II

GI Balandina.
Control system synthesis by means of cartesian genetic programming.
Procedia Computer Science, 103:176–182, 2017.

Augusto André Souza Berndt, Brunno Abreu, Isac S Campos, Bryan Lima, Mateus Grellert, Jonata T Carvalho, and Cristina Meinhardt.
A cgp-based logic flow: optimizing accuracy and size of approximate circuits.
Journal of Integrated Circuits and Systems, 17(1):1–12, 2022.

Camilo Andrés Cáceres Flórez, João Maurício Rosário, and Dario Amaya.
Control structure for a car-like robot using artificial neural networks and genetic algorithms.
Neural computing and applications, 32(20):15771–15784, 2020.

Janet Clegg, James Alfred Walker, and Julian Francis Miller.
A new crossover technique for cartesian genetic programming.
In Hod Lipson, editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007*, pages 1580–1587. ACM, 2007.

Kévin Cortacero, Brienne McKenzie, Sabina Müller, Roxana Khazen, Fanny Lafouresse, Gaëlle Corsaut, Nathalie Van Acker, François-Xavier Frenois, Laurence Lamant, Nicolas Meyer, et al.
Evolutionary design of explainable algorithms for biomedical image segmentation.
Nature communications, 14(1):7112, 2023.

Henning Cui, Michael Heider, and Jörg Hähner.
Positional bias does not influence cartesian genetic programming with crossover.
In Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, and Thomas Bäck, editors, *Parallel Problem Solving from Nature – PPSN XVIII*, pages 151–167, Cham, 2024. Springer Nature Switzerland.

References III

José Eduardo Henriques da Silva and Heder Bernardino.
Cartesian genetic programming with crossover for designing combinational logic circuits.
In *7th Brazilian Conference on Intelligent Systems, BRACIS 2018, São Paulo, Brazil, October 22-25, 2018*, pages 145–150. IEEE Computer Society, 2018.

Elsa Disdier, Rafael Almar, Rachid Benshila, Mahmoud Al Najar, Romain Chassagne, Debajoy Mukherjee, and Dennis G Wilson.
Predicting beach profiles with machine learning from offshore wave reflection spectra.
Environmental Modelling & Software, 183:106221, 2025.

Rongliang Fu, Robert Wille, and Tsung-Yi Ho.
Rcgp: An automatic synthesis framework for reversible quantum-flux-parametron logic circuits based on efficient cartesian genetic programming.
In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.

Cosijopii Garcia-Garcia, Alicia Morales-Reyes, and Hugo Jair Escalante.
Continuous cartesian genetic programming based representation for multi-objective neural architecture search.
Applied Soft Computing, 147:110788, 2023.

Brian W. Goldman and William F. Punch.
Analysis of cartesian genetic programming's evolutionary mechanisms.
IEEE Transactions on Evolutionary Computation, 19(3):359–373, 2015.

Radek Hrbacek and Vaclav Dvorak.
Bent function synthesis by means of cartesian genetic programming.
In *Parallel Problem Solving from Nature–PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings 13*, pages 414–423. Springer, 2014.

References IV

Tomas Hulka, Radomil Matousek, Ladislav Dobrovsky, Jakub Kudela, and Ondrej Hojny.
Cartesian genetic programming with a modified selection operator for combinational circuit design: Arithmetic multipliers and adders.
In *International Conference on Artificial Intelligence and Soft Computing*, pages 53–65. Springer, 2024.

Jakub Husa and Roman Kalkreuth.
A comparative study on crossover in cartesian genetic programming.
In Mauro Castelli, Lukás Sekanina, Mengjie Zhang, Stefano Cagnoni, and Pablo García-Sánchez, editors, *Genetic Programming - 21st European Conference, EuroGP 2018, Parma, Italy, April 4-6, 2018, Proceedings*, volume 10781 of *Lecture Notes in Computer Science*, pages 203–219. Springer, 2018.

T. Kalganova.
Evolutionary approach to design multiple-valued combinational circuits.
In *Proceedings. of the 4th International conference on Applications of Computer Systems (ACS'97)*, pages 333–339, Szczecin, Poland, 1997.

Roman Kalkreuth.
Two new mutation techniques for cartesian genetic programming.
In Juan Julián Merelo Guervós, Jonathan M. Garibaldi, Alejandro Linares-Barranco, Kurosh Madani, and Kevin Warwick, editors, *Proceedings of the 11th International Joint Conference on Computational Intelligence, IJCCI 2019, Vienna, Austria, September 17-19, 2019*, pages 82–92. ScitePress, 2019.

Roman Kalkreuth.
A comprehensive study on subgraph crossover in cartesian genetic programming.
In Juan Julián Merelo Guervós, Jonathan M. Garibaldi, Christian Wagner, Thomas Bäck, Kurosh Madani, and Kevin Warwick, editors, *Proceedings of the 12th International Joint Conference on Computational Intelligence, IJCCI 2020, Budapest, Hungary, November 2-4, 2020*, pages 59–70. SCITEPRESS, 2020.

References V

Roman Kalkreuth.
An Empirical Study on Insertion and Deletion Mutation in Cartesian Genetic Programming, pages 85–114. Springer International Publishing, Cham, 2021.

Roman Kalkreuth.
Phenotypic duplication and inversion in cartesian genetic programming applied to boolean function learning. In *Genetic and Evolutionary Computation Conference, Boston, USA, July 9-13, 2022, Companion Material Proceedings*, Genetic and Evolutionary Computation Conference 2022 (GECCO '22), New York, NY, USA, 2022. ACM.

Roman Kalkreuth.
Towards discrete phenotypic recombination in cartesian genetic programming. In Günter Rudolph, Anna V. Kononova, Hernán E. Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tusar, editors, *Parallel Problem Solving from Nature - PPSN XVII - 17th International Conference, PPSN 2022, Dortmund, Germany, September 10-14, 2022, Proceedings, Part II*, volume 13399 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2022.

Roman Kalkreuth.
Towards phenotypic duplication and inversion in cartesian genetic programming. In Thomas Bäck, Bas van Stein, Christian Wagner, Jonathan M. Garibaldi, H. K. Lam, Marie Cottrell, Faiyaz Doctor, Joaquim Filipe, Kevin Warwick, and Janusz Kacprzyk, editors, *Proceedings of the 14th International Joint Conference on Computational Intelligence, IJCCI 2022, Valletta, Malta, October 24-26, 2022*, pages 50–61. SCITEPRESS, 2022.

Roman Kalkreuth, Günter Rudolph, and Andre Droschinsky.
A new subgraph crossover for cartesian genetic programming. In James McDermott, Mauro Castelli, Lukás Sekanina, Evert Haasdijk, and Pablo García-Sánchez, editors, *Genetic Programming - 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*, volume 10196 of *Lecture Notes in Computer Science*, pages 294–310, 2017.

References VI

Roman Kalkreuth, Zdenek Vasícek, Jakub Husa, Diederick Vermetten, Furong Ye, and Thomas Bäck.
General boolean function benchmark suite. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA 2023, Potsdam, Germany, 30 August 2023 - 1 September 2023*, pages 84–95. ACM, 2023.

Roman Tobias Kalkreuth.
Reconsideration and Extension of Cartesian Genetic Programming. PhD thesis, 2021.

Paul Kaufmann and Roman Kalkreuth.
Parametrizing cartesian genetic programming: An empirical study. In Gabriele Kern-Isberner, Johannes Fürnkranz, and Matthias Thimm, editors, *KI 2017: Advances in Artificial Intelligence - 40th Annual German Conference on AI, Dortmund, Germany, September 25-29, 2017, Proceedings*, volume 10505 of *Lecture Notes in Computer Science*, pages 316–322. Springer, 2017.

Paul Kaufmann and Roman Kalkreuth.
On the parameterization of cartesian genetic programming. In *IEEE Congress on Evolutionary Computation, CEC 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–8. IEEE, 2020.

Paul Kaufmann and Marco Platzner.
Advanced techniques for the creation and propagation of modules in cartesian genetic programming. In Conor Ryan and Maarten Keijzer, editors, *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, pages 1219–1226. ACM, 2008.

SA Kazarlis, J Kalomiros, and V Kalaitzis.
A cartesian genetic programming approach for evolving optimal digital circuits. *Journal of Engineering Science & Technology Review*, 9(5), 2016.

References VII

Maryam Mahsal Khan, Arbab Masood Ahmad, Gul Muhammad Khan, and Julian F Miller.
Fast learning neural networks using cartesian genetic programming. *Neurocomputing*, 121:274–289, 2013.

Maryam Mahsal Khan, Gul Muhammad Khan, and Julian F Miller.
Evolution of neural networks using cartesian genetic programming. In *IEEE congress on evolutionary computation*, pages 1–8. IEEE, 2010.

Maryam Mahsal Khan, Gul Muhammad Khan, and Julian Francis Miller.
Evolution of optimal anns for non-linear control problems using cartesian genetic programming. In *IC-AI*, pages 339–346. Citeseer, 2010.

Maciej Krzywda, Szymon Łukasik, et al.
Cartesian genetic programming approach for designing convolutional neural networks. *arXiv preprint arXiv:2410.00129*, 2024.

Juergen Leitner, Simon Harding, Alexander Forster, and Jurgen Schmidhuber.
Mars terrain image classification using cartesian genetic programming, 2012.

Michael A Lones, Jane E Alty, Jeremy Cosgrove, Philippa Duggan-Carter, Stuart Jamieson, Rebecca F Naylor, Andrew J Turner, and Stephen L Smith.
A new evolutionary algorithm-based home monitoring device for parkinson's dyskinesia. *Journal of medical systems*, 41:1–8, 2017.

Marcus Märtens and Dario Izzo.
Neural network architecture search with differentiable cartesian genetic programming for regression. In *Proceedings of the genetic and evolutionary computation conference companion*, pages 181–182, 2019.

References VIII

Marcus Märtens and Dario Izzo.
Symbolic regression for space applications: Differentiable cartesian genetic programming powered by multi-objective memetic algorithms. *arXiv preprint arXiv:2206.06213*, 2022.

J. F. Miller, P. Thomson, and T. Fogarty.
Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 105–131. Wiley, 1997.

Julian F. Miller.
An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1135–1142, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

Julian F. Miller and Stephen L. Smith.
Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2):167–174, April 2006.

Julian F Miller and Peter Thomson.
A developmental method for growing graphs and circuits. In *International Conference on Evolvable Systems*, pages 93–104. Springer, 2003.

Julian F Miller, Dennis G Wilson, and Sylvain Cussat-Blanc.
Evolving developmental programs that build neural networks for solving multiple problems. *Genetic Programming Theory and Practice XVI*, pages 137–178, 2019.

References IX


Gul Muhammad Khan, Fahad Ullah, and Sahibzada Ali Mahmud.
Mpeg-4 internet traffic estimation using recurrent cgpann.
In *International Conference on Engineering Applications of Neural Networks*, pages 22–31. Springer, 2013.


HC Prashanth, Madhav Rao, T Bäck, B van Stein, C Wagner, JM Garibaldi, and HK Lam.
Improving digital circuit synthesis of complex functions using binary weighted fitness and variable mutation rate in cartesian genetic programming.
In *IJCCI*, pages 112–120, 2022.


Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao.
A genetic programming approach to designing convolutional neural network architectures.
In *Proceedings of the genetic and evolutionary computation conference*, pages 497–504, 2017.


Andrew James Turner and Julian Francis Miller.
The importance of topology evolution in neuroevolution: a case study using cartesian genetic programming of artificial neural networks.
In *International conference on innovative techniques and applications of artificial intelligence*, pages 213–226. Springer, 2013.


Andrew James Turner and Julian Francis Miller.
Recurrent cartesian genetic programming of artificial neural networks.
Genetic Programming and Evolvable Machines, 18:185–212, 2017.


Zdenek Vasicek and Lukas Sekanina.
Formal verification of candidate solutions for post-synthesis evolutionary optimization in evolvable hardware.
Genetic Programming and Evolvable Machines, 12:305–327, 2011.


Zdenek Vasicek and Lukas Sekanina.
Evolutionary approach to approximate digital circuits design.
IEEE Transactions on Evolutionary Computation, 19(3):432–444, 2014.

References X


Vesselin K. Vassilev and Julian F. Miller.
The advantages of landscape neutrality in digital circuit evolution.
In Julian Miller, Adrian Thompson, Peter Thomson, and Terence C. Fogarty, editors, *Evolvable Systems: From Biology to Hardware*, pages 252–263, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.


Tina Yu and Julian Miller.
Neutrality and the evolvability of boolean function landscape.
In Julian Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G. B. Tettamanzi, and William B. Langdon, editors, *Genetic Programming*, pages 204–217, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.


Faheem Zafari, Gul Muhammad Khan, Mehreen Rehman, and Sahibzada Ali Mahmud.
Evolving recurrent neural network using cartesian genetic programming to predict the trend in foreign currency exchange rates.
Applied Artificial Intelligence, 28(6):597–628, 2014.