General Boolean Function Benchmark Suite

Chair of Artificial Intelligence Methodology, RWTH Aachen

Tuesday 12th December, 2023

Dr. rer. nat. Roman Kalkreuth

General Boolean Function Benchmark Suite

Roman Kalkreuth
r.t.kalkreuth@liacs.leidenuniv.nl
Computer Lab of Paris 6, Sorbonne University
Paris, France

Zdeněk Vašíček
vasicek@fit.vut.cz
Brno University of Technology
Brno, Czech Republic

Jakub Husa
ihusa@fit.vut.cz
Brno University of Technology
Brno, Czech Republic

Diederick Vermetten
d.l.vermetten@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science, Leiden University
Leiden, Netherlands

Furong Ye
f.ye@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science, Leiden University
Leiden, Netherlands

Thomas Bäck
T.H.W.Baeck@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science, Leiden University
Leiden, Netherlands

[1] Kalkreuth et al.: *General Boolean Function Benchmark Suite*, FOGA'23: Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, Potsdam (Germany), 2023

Background

- Lots of machine learning models such as LLM's require pre-training
- Genetic Programming $\rightarrow$ learning from scratch
- Search on high-level symbolic representations of problems $\rightarrow$ human-readable solutions

Taxonomy

- Genetic Programming is a search heuristic
- Evolutionary algorithm-based method for the synthesis of computer programs
- Inspired by Charles Darwin's theory of evolution[2]

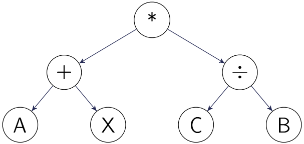
[2] Charles Darwin: *On the Origin of Species by Means of Natural Selection*, 1859

SCIENCES SORBONNE UNIVERSITÉ

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**.



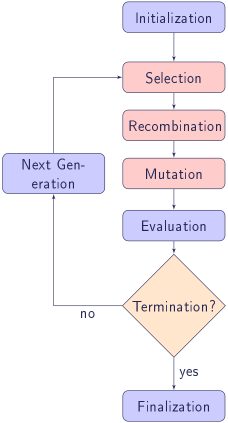
$$\mathcal{F} = \{ +, -, *, \div \}$$

$$\mathcal{T} = \{ A, X, C, B \}$$

$$\mathcal{E} = \text{Edges}$$

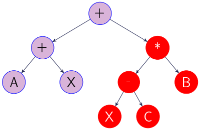
$$\Psi = (A + X) * (C \div B)$$

Genetic Programming (GP)

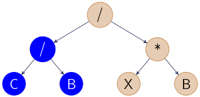


Termination criteria → predefined fitness reached or budget of generations exceeded

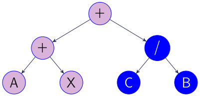
(a) First Parent



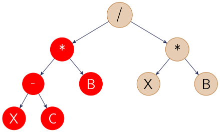
(b) Second Parent



(c) First Offspring

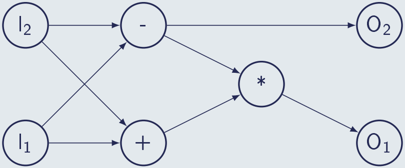


(d) Second Offspring



Background

- Genetic Programming → traditionally tree representation
- Cartesian Genetic Programming → graph representation
- Extension of Genetic Programming



Representation

- Program representation $\rightarrow$ acyclic and directed graph
- Genotype-phenotype mapping $\rightarrow$ encoding-decoding of the graph
- Predominantly used without recombination in the past $\rightarrow$ $(1+\lambda)$ selection scheme (*"vanilla CGP"*)
- Recombination operators have been proposed in recent years[3]

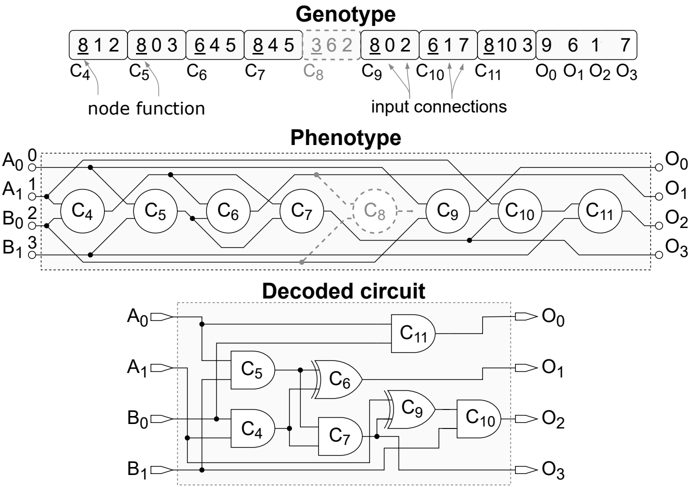
[3] Roman Kalkreuth: *Towards Discrete Phenotypic Recombination in Cartesian Genetic Programming*, International Conference on Parallel Problem Solving from Nature (PPSN'22), Dortmund (Germany), 2022

Definition (Cartesian Genetic Program)

A cartesian genetic program is an element of the Cartesian product $\mathcal{N}_\mathrm{i} \times \mathcal{N}_\mathrm{f} \times \mathcal{N}_\mathrm{o} \times \mathcal{F}$:

- $\mathcal{N}_\mathrm{i}$ is a finite non-empty set of input nodes
- $\mathcal{N}_\mathrm{f}$ is a finite set of function nodes
- $\mathcal{N}_\mathrm{o}$ is a finite non-empty set of output nodes
- $\mathcal{F}$ is a finite non-empty set of functions

Cartesian Genetic Programming (CGP)



Vanilla CGP

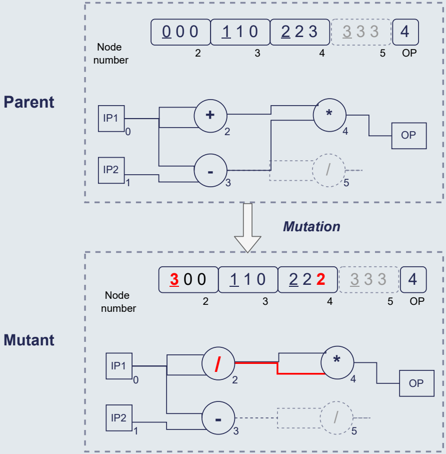
Algorithm 1 $1+\lambda$) evolutionary strategy

repeat
  initialize(P) ▷ *Initialize parent individual*
  $Q \leftarrow$ breed(P) ▷ *Breed λ offspring by mutation*
  Evaluate(Q) ▷ *Evaluate the fitness of the offspring*
  if *at least one individual of* Q *has better fitness then* P **then**
    $P \leftarrow$ best(Q) ▷ *Replace the parent by the best offspring*
  end if
until P *meets termination criterion*
return P

Mutation

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

Example of probabilistic point mutation in CGP



Background

- Logic synthesis (LS) in GP can be considered a **black-box** and **optimization** problem domain
- Synthesis of **Boolean expressions** that match the **input-output mapping** of Boolean functions
- Logic synthesis from scratch → LSZero
- Real world application → implementation as digital circuits

Definition (Boolean Function)

- A Boolean function is a mathematical function where the values of the respective arguments and the result vary in a set with two elements (usually $\{0, 1\}$ or $\{\mathrm{true}, \mathrm{false}\}$)
- A Boolean function of degree n with one output is defined as $f : \{0, 1\}^n \mapsto \{0, 1\}$
- A Boolean function with multiple outputs as $f : \{0, 1\}^n \mapsto \{0, 1\}^m$ where $m > 1$.

Boolean Function (Example)

Inputs		Outputs	
I1	I2	Q1	Q2
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	1

Definition (Boolean Expression)

- A Boolean expression is an algebraic expression of a Boolean function that can be a finite combination of Boolean constants, variables, and logical operators.
- The evaluation of a Boolean expression leads to a Boolean value.

Boolean Expression (Example)

$Q_1 = \mathsf{I}_1 \wedge \mathsf{I}_2 \qquad Q_2 = \mathsf{I}_1 \vee \mathsf{I}_2 \qquad \mathcal{E} := \{Q_1, Q_2\}$

SCIENCES SORBONNE UNIVERSITÉ

Evaluation

- Cost function → Hamming distance
- Hamming distance measures the difference between the desired and actual outputs of a candidate circuit

Definition (Hamming Distance)

Let Σ be a finite alphabet and $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ two words from Σ^n of length n. The Hamming distance between x and y is defined as:
$\Delta(x, y) := \left| \{ i \in \{1, \ldots, n\} \mid x_i \neq y_i \} \right|$

Hamming Distance (Example)

$x = (1, 0, 0, 1, 0) \quad y = (0, 0, 1, 1, 0) \quad \Delta(x, y) := 2$

Background and Motivation

- 2012: First comprehensive review on the state of benchmarking in GP[4]
- Mismatch between problems used to test the performance of GP systems and real-world problems
- Several benchmark suites in major GP problem domains have been proposed since 2012
    - A general benchmark suite for LS is still missing

[4] McDermott et al.: *Genetic programming needs better benchmarks*, Genetic and Evolutionary Computation Conference, Philadelphia, Pennsylvania (USA) (GECCO'12), 2012

Main Objectives and Properties

- **Generalization** $\rightarrow$ covering a broad spectrum of types of Boolean functions.
- **Scalability** $\rightarrow$ proposing benchmarks of the same type with different bit lengths of the respective inputs.
- Scaling up the bit length increases the difficulty of the proposed benchmarks but maintains the similarity

Problem selection

- Thoughtfully selected Boolean functions from seven popular categories: **arithmetic**, **transmission**, **comparison**, **counting**, **mixed**, **parity** and **cryptography**.
- High dimensional multiple-output functions that remarkably differ from single-output benchmarks
- Synthesis of cryptographic Boolean functions depicts a comparatively young subfield of LS

Abbreviations, names, and types of the selected benchmark functions.

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

Function set of the arithmetic logic unit benchmark

Opcode	Function	Description
000	&	Logical and
001	$\|\|$	Logical or
010	$\oplus$	Exclusive or
011	$+$	Addition
100	$-$	Subtraction

SCIENCES SORBONNE UNIVERSITÉ

Baseline results

- To obtain baseline results, we used CGP with the $(1 + \lambda)$ evolutionary strategy.
- We performed 100 runs and measured the number of function evaluations
- A limit of 10^8 function evaluations was defined in each run

General Boolean Function Benchmark Suite (GBFS)

Problem			Search Performance Evaluation					
Identifier	Inputs count	Output count	Average	Standard deviation	Q1	Median	Q3	Success rate
add4	9	5	1,176,250	1,241,222	559,810	864,099	1,244,870	100%
add6	13	7	4,354,846	3,425,693	2,098,854	3,358,271	4,883,583	100%
add8	17	9	15,118,552	12,166,477	6,549,025	11,308,473	18,063,144	100%
mul3	6	6	439,106	337,382	229,458	317,891	524,669	100%
mul4	8	8	37,074,573	16,213,144	23,682,377	36,943,707	47,053,177	100%
mul5	10	10	-	-	-	-	-	0%
dec4	4	2	11,520	11,199	4,722	8,326	12,079	100%
dec8	8	3	656,502	612,189	277,681	449,462	826,220	100%
dec16	16	4	16,657,577	9,659,375	9,822,953	13,514,673	20,891,128	100%
enc8	3	8	22,895	12,751	13,383	20,015	29,372	100%
enc16	4	16	133,268	55,065	90,761	125,830	163,579	100%
enc32	5	32	6,048,000	5,253,472	1,669,344	4645,978	8,446,886	100%
icomp5	5	30	298,378	253,280	148,066	231,331	351,607	100%
icomp7	7	63	1,476,305	771,096	906,376	1,318,098	1,812,708	100%
icomp9	9	108	5,472,300	2,475,782	3,761,209	5,102,597	6,401,704	100%
mcomp4	8	3	1,418,028	951,804	733,138	1,033,987	2,051,997	100%
mcomp5	10	3	2,898,336	1,923,573	1,593,730	2,489,199	3,416,328	100%
mcomp6	12	3	7,975,125	5,560,237	4,075,013	6,523,368	10,284,085	100%
count4	4	3	44,626	42,380	19,567	30,017	47,265	100%
count6	6	4	853,994	890,100	368,770	649,935	1,108,540	100%
count8	8	4	8,506,276	8,519,358	2,992,012	6,490,877	9,768,313	100%
count10	10	5	29,674,777	17,411,768	16,582,947	26,897,332	40,192,421	86%
alu4	11	5	11,342,677	5,995,435	6,961,372	10,437,724	14,009,597	100%
alu6	15	7	45,341,485	20,465,133	26,869,723	42,071,437	59,961,675	93%
alu8	19	9	72,811,646	16,498,376	63,702,578	74,942,768	85,688,638	23%
epar8	8	1	2,215,101	1,436,094	1,147,611	1,877,298	2,709,974	100%
epar9	9	1	4,232,008	2,935,317	2,130,972	3,375,780	5,159,903	100%
epar10	10	1	8,155,620	6,950,750	3,518,499	5,847,038	10,574,672	100%
epar11	11	1	11,869,051	9,964,340	6,123,499	8,183,564	13,465,052	100%
bent8	8	1	1,831	4,736	514	881	1,504	100%
bent12	12	1	3,262	2,764	1,543	2,514	4,045	100%
bent16	16	1	8,128	8,161	2,867	5,219	10,067	100%
bal8	8	1	103,807	124,303	27,635	68,280	143,775	100%
bal12	12	1	394,556	438,565	101,052	218,533	543,450	100%
res8	8	1	16,712	40,763	3,849	7,062	13,405	100%
res12	12	1	170,153	178,829	39,825	100,651	248,155	100%
mask8	8	1	3,406	3,694	1,536	2,481	3,919	100%
mask12	12	1	59,812	60,904	23,446	38,928	75,591	100%

Analysis of Benchmark Complexity

- Proposed benchmarks have different features
- Using multiple metrics to analyze the complexity accurately and holistically
- Properties of the truth table, normal forms and multi-level gate logic

General Boolean Function Benchmark Suite (GBFS)

Problem			Truth table metrics			Representation: Normal forms & ROBDD			Representation: Multi-level			
Identifier	Input count	Output count	Function symmetry	Hamming weight	Avalanche effect	DNF terms	ANF terms	BDD nodes	AIG nodes	AIG depth	MAP gates	MAP depth
add4	9	5	46%	50%	1.8	135	65	43	39	10	44	8
add6	13	7	40%	50%	1.8	607	259	82	61	13	68	10
add8	17	9	36%	50%	1.9	2,519	1,029	133	87	15	97	12
mul3	6	6	42%	29%	1.7	35	27	46	37	11	42	9
mul4	8	8	37%	33%	2.4	145	138	145	83	16	89	13
mul5	10	10	34%	36%	2.9	591	671	440	146	21	157	17
dec4	4	2	0%	69%	0.6	4	8	5	3	2	5	3
dec8	8	3	21%	80%	0.4	12	151	17	12	6	21	7
dec16	16	4	27%	85%	0.2	32	39,062	49	35	14	62	15
enc8	3	8	67%	13%	2.0	8	20	24	12	2	38	8
enc16	4	16	75%	6%	2.0	16	66	64	24	2	109	9
enc32	5	32	65%	3%	2.0	32	212	160	110	9	267	11
icomp5	5	30	40%	33%	8.0	40	50	60	30	2	45	2
icomp7	7	63	29%	33%	12.0	84	105	126	63	2	91	2
icomp9	9	108	22%	33%	16.0	144	180	216	108	2	153	2
mcomp4	8	3	36%	33%	0.6	46	161	33	20	7	40	7
mcomp5	10	3	33%	33%	0.5	94	485	42	25	7	49	9
mcomp6	12	3	32%	33%	0.4	190	1,457	51	31	9	61	9
count4	4	2	100%	56%	1.5	15	10	10	14	7	13	4
count6	6	3	100%	47%	1.8	73	36	28	26	12	33	7
count8	8	3	100%	53%	1.8	293	106	42	46	16	49	9
count10	10	4	100%	45%	1.8	1,228	310	80	76	21	84	12
alu4	11	5	32%	46%	1.4	170	240	81	84	12	125	10
alu6	15	7	29%	47%	1.5	740	1,434	157	132	14	183	12
alu8	19	9	28%	48%	1.6	3,038	10,980	257	184	16	252	14
epar8	8	1	100%	50%	1.0	128	8	8	21	6	14	3
epar9	9	1	100%	50%	1.0	256	9	9	24	8	16	4
epar10	10	1	100%	50%	1.0	512	10	10	27	8	18	4
epar11	11	1	100%	50%	1.0	1,024	11	11	30	8	20	4

Conclusions

- We proposed a diverse benchmark suite for logic synthesis, consisting of seven distinct categories that represent problems of varying levels of difficulty
- Our suite can support the evaluation of the search performance and robustness of GP methods in evolutionary-driven synthesis of Boolean functions
- We also provided an overview of the complexity characteristics of our proposed benchmarks which can be used for further analysis

SCIENCES SORBONNE UNIVERSITÉ

GitHub Repository

The benchmark files and corresponding interfaces for Java, C++ und Python are available at:
https://github.com/boolean-function-benchmarks

[1] Roman Kalkreuth et al. "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*. ACM, 2023, pp. 84–95. DOI: 10.1145/3594805.3607131. URL: https://doi.org/10.1145/3594805.3607131.