# Towards a General Boolean Function Benchmark Suite

Roman Kalkreuth[⋆], Zdeněk Vašíček[†], Jakub Husa[†], Diederick Vermetten[‡], Furong Ye[‡] and Thomas Bäck[‡]

[⋆]Sorbonne University (Paris, France), [†]Brno University of Technology (Brno, Czech Republic), [‡]Leiden University (Leiden, The Netherlands)

## Abstract

- Just over a decade ago, the first review on benchmarking in Genetic Programming (GP) stated that GP needs better benchmarks [1].
- Several benchmark suites in major GP problem domains have been proposed over time.
- A diverse and accessible benchmark suite for logic synthesis is still missing.
- We take a first step towards a benchmark suite that covers different types of Boolean functions.

## Logic Synthesis in Genetic Programming

- Logic synthesis (LS) in GP can be considered a black-box and optimization problem domain.
- LS by means of GP refers to the synthesis of expressions that match the input-output mapping of Boolean functions.
- LS played a major role in the application scope of GP research throughout its history.

## General Motivation

- A general benchmark suite for LS is still missing in the field of GP.
- Promotion of LS to maintain a vivid application scope in GP.
- Cultivating diversity and accessibility in this problem domain.

## Main Objectives and Properties

- The philosophy behind our proposed benchmark suite respects the following objectives and properties: **Generalization**, **Accessibility** and **Scalability**.
- A generalized approach to benchmarking in LS is achieved by covering a broad spectrum of types of Boolean functions.
- We make use of the scaling property of Boolean functions by 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 similarity.

## Problem Selection

- We thoughtfully selected Boolean functions from seven popular categories: **arithmetic**, **transmission**, **comparison**, **counting**, **mixed**, **parity** and **cryptography**.
- We selected high dimensional multiple-output functions which remarkably differ from the overused single-output parity and multiplexer benchmarks.
- The synthesis of cryptographic Boolean functions depicts a comparatively young and interesting field of application in LS and it is therefore natural to propose these functions as benchmarks.

## Interfaces and Resources

- Data files for the benchmarks are available in several formats and publicly accessible in our GitHub repository.
- Interfaces for C++, Java and Python can also be found in our repository.

## Future Work

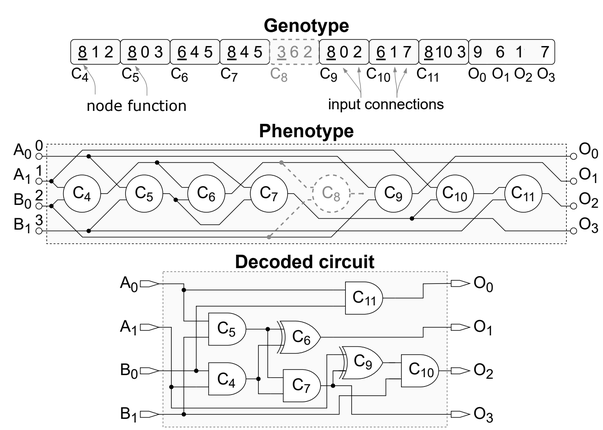
- In a **forthcoming full technical paper** we will propose our benchmark suite.
- The paper will be published in the framework of the **ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA)** this year.
- We also analyze the complexity of the proposed benchmarks in this paper.

*List of Boolean functions that have been selected for our benchmark suite.*

| Problem name | Identifier | # Inputs | # Outputs |
|---|---|---|---|
| **Arithmetic** | | | |
| Adder 4-Bit | add4 | 7 | 4 |
| Adder 6-Bit | add6 | 9 | 5 |
| Adder 8-Bit | add8 | 11 | 6 |
| Multiplier 3-Bit | mul3 | 6 | 6 |
| Multiplier 4-Bit | mul4 | 8 | 8 |
| Multiplier 5-Bit | mul5 | 10 | 10 |
| **Transmission** | | | |
| One-hot decoder 4-Bit | dec4 | 4 | 2 |
| One-hot decoder 8-Bit | dec8 | 8 | 3 |
| One-hot decoder 16-Bit | dec16 | 16 | 4 |
| One-hot decoder 8-Bit | enc8 | 3 | 8 |
| One-hot decoder 16-Bit | enc16 | 4 | 16 |
| One-hot decoder 32-Bit | enc32 | 5 | 32 |
| **Comparative** | | | |
| Identity Comparator 5-Bit | icomp4 | 4 | 18 |
| Identity Comparator 7-Bit | icomp5 | 5 | 30 |
| Identity Comparator 9-Bit | icomp6 | 6 | 45 |
| Magnitude Comparator 4-Bit | mcomp4 | 8 | 3 |
| Magnitude Comparator 5-Bit | mcomp5 | 10 | 3 |
| Magnitude Comparator 6-Bit | mcomp5 | 12 | 3 |
| **Counting** | | | |
| Ones' counter 4-Bit | count4 | 4 | 3 |
| Ones' counter 6-Bit | count6 | 6 | 4 |
| Ones' counter 8-Bit | count8 | 8 | 4 |
| Ones' counter 10-Bit | count10 | 10 | 5 |
| **Mixed (building blocks)** | | | |
| Arithmetic Logic Unit 4-Bit | alu4 | 11 | 5 |
| Arithmetic Logic Unit 6-Bit | alu6 | 15 | 7 |
| Arithmetic Logic Unit 8-Bit | alu8 | 19 | 9 |
| **Parity** | | | |
| Parity-even 8-Bit | epar8 | 8 | 1 |
| Parity-even 9-Bit | epar9 | 9 | 1 |
| Parity-even 10-Bit | epar10 | 10 | 1 |
| Parity-even 11-Bit | epar11 | 11 | 1 |
| **Cryptographic** | | | |
| Bent 8-Bit | Ben8 | 8 | 1 |
| Bent 12-Bit | Ben12 | 12 | 1 |
| Bent 16-Bit | Ben12 | 16 | 1 |
| Balanced 8-Bit | Bal8 | 8 | 1 |
| Balanced 12-Bit | Bal12 | 12 | 1 |
| Resilient 8-Bit | Res8 | 8 | 1 |
| Resilient 12-Bit | Res12 | 12 | 1 |
| Masking 8-Bit | Mas8 | 8 | 1 |
| Masking 12-Bit | Mas12 | 12 | 1 |

## Baseline results

- Baseline results have been obtained with Cartesian Genetic Programming (CGP) [2].
- CGP is a graph-based representation model for GP that can be used for automated circuit design (see figure below).
- Results are available in our GitHub repository.




*CGP encodes an acyclic and directed graph that can be used to represent a candidate circuit.*

## References

[1] James McDermott, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaskowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong, and Una-May O'Reilly. Genetic programming needs better benchmarks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, page 791–798, New York, NY, USA, 2012. Association for Computing Machinery.

[2] Julian F. Miller and Peter Thomson. Cartesian genetic programming. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin, and Terence C. Fogarty, editors, *Genetic Programming, European Conference, Edinburgh, Scotland, UK, April 15-16, 2000, Proceedings*, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2000.