Evolving Parsimonious Circuits through Shapley Value-based Genetic Programming

Xinming Shi
University of Birmingham, UK
Southern University of Science and Technology, China
xxs972@student.bham.ac.uk

Jiashi Gao
Southern University of Science and Technology, China
12131101@mail.sustech.edu.cn

Leandro L. Minku, Xin Yao*
University of Birmingham, UK
l.l.minku@bham.ac.uk
University of Birmingham, UK
Southern University of Science and Technology, China
xiny@sustech.edu.cn

ABSTRACT

Evolutionary analog circuit design is a challenging task due to the large search space incurred by the circuit topology and device values. Applying genetic operators on randomly selected genes may make it difficult to identify which part of sub-circuit is beneficial to the evolution and even destroy useful sub-circuits, potentially incurring stagnation of the evolutionary process and bloat on the evolved circuits. In this paper, we propose a tree-based approach called Shapley Circuit Tree that incorporates Shapley values for quantifying the contribution of each function node of the circuit tree to the performance of the whole tree, to guide the evolutionary process. Our experiments on three benchmarks show that the proposed approach is able to evolve analog circuits with smaller area while converging faster than existing approaches.

KEYWORDS

Evolutionary analog circuit design, tree-based circuit representation, Shapley value, genetic programming, evolvable hardware

ACM Reference Format:
Xinming Shi, Jiashi Gao, and Leandro L. Minku, Xin Yao. 2022. Evolving Parsimonious Circuits through Shapley Value-based Genetic Programming. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3529032

1 INTRODUCTION

Evolutionary circuit design has been attracting increasing attention, with analog circuit design being particularly challenging due to its complex topology and parameter selection [6]. To better evolve analog circuits in terms of the circuit size and search efficiency, two key factors are taken into account when designing an evolutionary method: circuit representation and genetic operators.

The hierarchical feature of tree representation was shown to be desirable for encoding analog circuits, since it naturally matches the hierarchy of function modules, e.g., sub-circuits in a circuit [10]. Genetic operators such as crossover and mutation are usually representation-dependent and are commonly applied to random genes [5]. However, randomly choosing genes to go through genetic operations may limit the search efficiency by loosing potentially useful sub-circuits. Moreover, due to the presence of devices with zero contribution, the bloat issue may occur, increasing the size of evolved circuit. If it was possible to identify which part of sub-circuit is beneficial to the evolution, the search efficiency could potentially be improved by using this knowledge to guide the optimisation process. Leave-one-out (LOO) [2] is an existing value evaluation approach that evaluates the contribution of each gene to the whole chromosome. However, it is not suitable for circuit design, because it would ignore the dependencies between devices/sub-circuits in the circuit. Therefore, a more suitable measure to evaluate the contribution of each gene in the circuit representation is desirable.

In this work, we propose a novel evolutionary framework that guides the optimisation process based on Shapley values as measures to identify which parts of the circuit are most beneficial when using a tree hierarchy circuit representation, leading to better evolved circuits. Shapley values can be understood as the weighted marginal contributions of each player to potential teams of players in cooperative game theory and economics [9] [8]. To the best of our knowledge, our proposed approach is the first algorithm that can compute the Shapley values for circuit devices or blocks.

Our key contributions are as follows:

(1) We propose an approach to evaluate the importance of the nodes in a circuit tree based on Shapley values.
(2) We develop a two-stage evolutionary framework based on the Shapley tree evaluation to guide the evolution of analog circuits towards more promising regions of the search space.
(3) We experimentally verify that our proposed approach can provide both faster evolutionary convergence and more parsimonious circuits.

2 SHAPLEY VALUE-BASED CIRCUIT EVOLUTION

2.1 Shapley-based tree and subtree evaluation

Consider the circuit tree representation proposed in our previous work [10]. For a tree T containing a function node set N_F, in order to identify the contribution of each function node $node_i \in N_F$ to the overall performance $V(T)$ in a circuit-plausible way, we need a suitable measurement method. This measurement denoted as

*Corresponding authors

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3529032

u should satisfy the following properties, where $SubTree(\cdot)$ is a function converting a node to its sub-tree:

(1) **Zero contribution:** In the circuit evolution, one decision to make is how to handle circuit devices/sub-circuits that have no contribution. Therefore, the measure u should be capable of identifying these devices/subcircuits. If $node_i \in N_F$ has no effect on performance when combined with any other $node_j$, it should be assigned a zero value. More precisely, for $\forall node_j \in N \backslash node_i$, if $V(SubTree(node_i) \cup SubTree(node_j)) = V(SubTree(node_j))$, $u_i = 0$;

(2) **Symmetry:** In the circuit evolution, there are some circuit devices/subcircuits that generate the same change in the performance. These could be referred to as symmetric elements. The measure should be capable of identifying these devices/subcircuits. If $node_i$ and $node_j$ always generate the same change in the performance when combined with any other $node_k$, then $node_i$ and $node_j$ should be assigned the same value by symmetry. More precisely, for $node_i$, $node_j$ and $\forall node_k \in N_F \backslash \{node_i, node_j\}$, if $V(SubTree(node_k) \cup SubTree(node_i)) = V(SubTree(node_k) \cup SubTree(node_j))$, $u_i = u_j$;

(3) **Additivity:** To evaluate the contribution of a subcircuit, we need to be able to define its contribution as the sum of the separate contributions of each of its devices. More precisely, for a function node subset S, if the overall fitness $V(SubTree(S))$ is the sum of each separate performance $V(node_i), node_i \in S$, the value u_S should be the sum of its value for each $node_i \in S$:

$$u_S = \sum_{i=1}^{|S|} u_{node_i}. \tag{1}$$

The Shapley formula shown in Eq. 2 uniquely satisfies all these properties:

$$u_{node_i} = \frac{1}{|N_F|} \sum_{S \subseteq N_F \backslash \{i\}} \left(V(SubTree_{S \cup node_i}) - V(SubTree_S) \right) / \binom{|N_F| - 1}{|S|} \tag{2}$$

We write $V(S)$ to denote the performance of subtree S. In our context, u_{node_i} is the Shapley value representing the contribution of the subtree rooted by $node_i$. The Shapley formula in Eq. 2 uniquely provides an equitable assignment of values to nodes. Computing Shapley, however, requires computing all the possible marginal contributions which is exponentially large in the node number. Here, we introduce a truncated Monte Carlo Shapley [3] for circumventing this problem. First, we sample a random permutation of function nodes and generate the sub-tree set π. Then, we scan the permutation from the first to the last element and calculate the marginal contribution of each element. The scan process is truncated when the fitness V_i of subtree i is within a pre-defined performance tolerance of the overall fitness $V(T)$ and sets the marginal contribution to zero for the rest of the elements in this permutation. By repeating this same procedure multiple times until a convergence criterion is met, the final result gives an unbiased estimate of the Shapley value. The pseudo code for the truncated Monte Carlo Shapley value procedure for a circuit tree is given in Algorithm 1. The convergence criterion is set as the maximum number of iterations.

Algorithm 1 Pseudocode of Truncated Monte Carlo Shapley of a circuit tree

```
def shapley_value(T : tree):
N_F ← function node set of T
Initialize v_i = 0 for i = 1, ..., |N_F|
while Convergence criteria not met do do
    t ← t + 1
    π : Random permutation of sub-trees with function node ∈ N_F as root
    V_0^t ← V(∅)
    for j ∈ {1, ..., |N_F|} do
        if |V(T) − V_{j−1}^t| < Performance Tolerance then
            V_j^t = V_{j−1}^t
        else
            V_j^t = V({π_1^t ∪ ... ∪ π_j^t})
        end if
        u_{π_j^t} ← ((t−1)/t) u_{π_j^{t−1}} + (1/t)(V_j^t − V_{j−1}^t)
    end for
end while
Return v_i
```

The calculation process of $V(s)$ is described in the following, where we consider two scenarios: the subtree π_j has the overlapped nodes with the subtrees $\pi_i \in \pi, i < j$, which means they are connected in their corresponding circuit counterpart; (2) the subtree π_j has no the overlapped nodes with the subtrees $\pi_i \in \pi, i < j$, which means they have no connection in their corresponding circuit counterpart. Their corresponding evaluation strategies are proposed as follows:

- **The subtree π_j has the overlapped nodes with the subtrees $\pi_i \in \pi, i < j$:** For $\pi = \{\pi_j\}$ $(j = 1, ..., n)$, the performance of $\{\pi_1 \cup ... \cup \pi_j\}$ can be calculated by:

$$V(\{\pi_1 \cup ... \cup \pi_j\}) = f(Vol_{H_q(\{\pi_1 \cup ... \cup \pi_j\})}, Vol_{Target}) - f(Vol_{H_1(\{\pi_1 \cup ... \cup \pi_j\})}, Vol_{Target}), j = 1, ..., n, \tag{3}$$

where $f(\cdot)$ is the fitness function, Vol indicates the voltage of one circuit node. Here we use voltage as the circuit measurement. And Vol_{Target} denotes the target voltage of the evolutionary circuit design. Based on the circuit tree formulation in our previous work [10], $H_q(\{\pi_1 \cup ... \cup \pi_j\})$ represents the last terminal node of a function node, where q is the last terminal of a function node. We usually apply this node as the output terminal of subtree π_j. $H_1(\{\pi_1 \cup ... \cup \pi_j\})$ represents the first terminal node of a function node. We usually apply this node as the input terminal of subtree π_j. Therefore, the first term on the right side of equal sign indicates the differences between the voltage on the input terminal of π and the target voltage, and the second term on the right side of equal sign indicates the differences between the voltage on the output terminal of π and the target voltage.

The difference of these two terms represents how much this sub-circuit decoded by π contributes towards achieving the target voltage.

- **The subtree π_j has no overlapped nodes with the subtrees $\pi_i \in \pi, i < j$:** For $\pi = \{\pi_j\}$ $(j = 1, ..., n)$, if the circuit counterpart of π_j is not connected with that of $\{\pi_1 \cup ... \cup \pi_{j-1}\}$, its performance can be calculated by:

$$V(\{\pi_1 \cup ... \cup \pi_j\}) = max(V(\{\pi_1 \cup ... \cup \pi_{j-1}\}), V(\pi_j)), \quad j = 2, ..., n. \tag{4}$$

Combined with Algorithm 1, under the calculation scheme of the *max* function in Eq. 4, the subtree π_j which cannot bring performance improvement compared with the subtrees $\{\pi_i, i < j\}$, will get $V_j - V_{j-1} = 0$. So, the increment of u_j will be zero, which means the contribution of π_j is also zero.

2.2 Shapley-based genetic programming

The overall flowchart of evolving analog circuits is shown in Fig. 1. The first step (parameter settings) sets the population size N, tournament size T, max iterations $Max\_Iteration$, shapley iterations $Shapley\_Iteration$, max depth of tree D, topology crossover rate P_{cross}, mutation rate P_{mutate} and value crossover rate $P_{valuecross}$. The population is initialized based on the tree hierarchy representation. Then, the initialized tree individuals are transformed into the circuit netlist, based on which the circuit is simulated. The fitness of each individual in the population is then evaluated based on the results of circuit simulation. According to the elitism strategy, the individual with the best fitness is saved in $P^{new}[0]$.

After that, the evolution has two possible stages. If the number of iterations is within a threshold $Shapley\_iter$, the first evolutionary stage is triggered (orange box). To stimulate diversity, this stage applies genetic operators (topology crossover, value crossover, and mutation) to random nodes as in previous work [10]. If the number of iterations is above $Shapley\_iter$, the second evolutionary stage is triggered (green box). During this stage, the Shapely-oriented crossover and mutation operations are applied. As for the Shapely-oriented crossover, the subtrees rooted by the function node with the best Shapley value in Parent 2 will be taken the crossover with the subtree rooted by the function node with the worst Shapley value in Parent 1, and their resulting child tree will be added to the next generation. As for the Shapely-oriented mutation operators, the function node with the worst Shapley value will be taken either the *delete* or *add* mutation operation. Therefore, this stage adopts Shapley-guided genetic operators to guide the evolutionary process towards more promising regions of the search space.

Both in the first and second stages, two parents are selected by n-tournament selection to go through topology crossover [10] with probability P_{cross}. And by the probability $P_{valuecross}$, the individuals will taken the value crossover operation [10], where it can only be executed to two function nodes of the same type of device.

3 EXPERIMENTS

The population size denoted as pop_Size setting as 100, $shapley_iter$ denotes the generations that will enter the stage of Shapley-oriented circuit evolution, which is set as 250, and the maximum iterations is set as 500. The rate of mutation, crossover and value crossover are set as 0.2, 0.8, 0.2, respectively. 10 runs are applied for the average results. These parameter values were adjusted experimentally from preliminary runs.

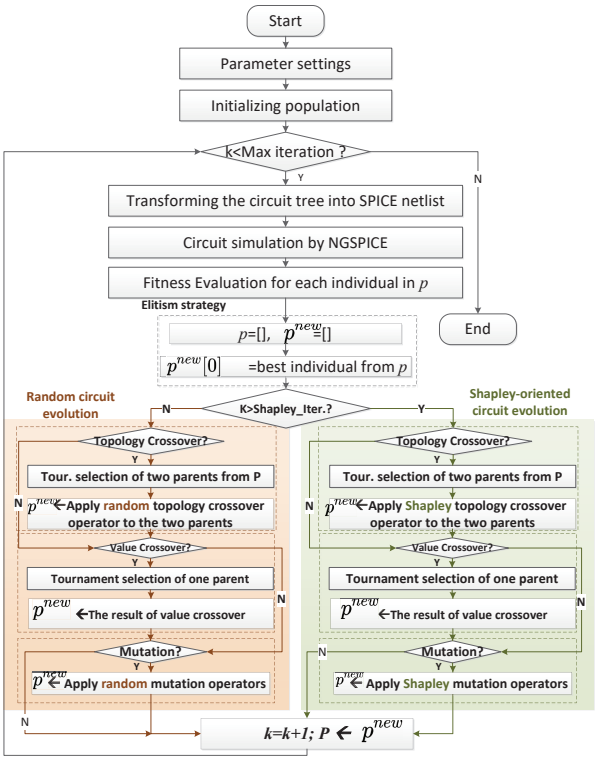

Figure 1: Overall flowchart of evolving analog circuits.

3.1 Shapley tree-based circuit evolution leads to faster convergence

The proposed method is evaluated on three benchmarks, namely voltage reference circuit, temperature sensor circuit, and Gaussian function generator. These benchmark circuits are widely applied to evaluate the evolutionary analog circuit design [1, 4, 7].

We perform an ablation study of Shapley circuit tree method on the stage 2 of the evolution. Specifically, the Shapley circuit tree method on the stage 2 is ablated, remaining the same as stage 1. Fig. 2 (top row) shows the fitness comparisons with and without Shapley on stage 2 (the approaches are equivalent during stage 1). According to the fitness comparisons, after 250 generations of stage 2, the fitness based on the evolution with Shapley values can converge to better values faster than the one without Shapley. The absolute best fitness ($|BF|$) and the average best fitness ($|MBF|$) for different tasks with and without Shapley are given in Table 1. As we can see, within Max_iter.=500, the evolution with Shapley can obtain better fitness compared with the one without Shapley. Combined with stimulating the diversity during the first stage evolution, the Shapley-oriented evolution will guide the genetic operations towards more promising search space, which may accelerate the process of pursuing the better individual.

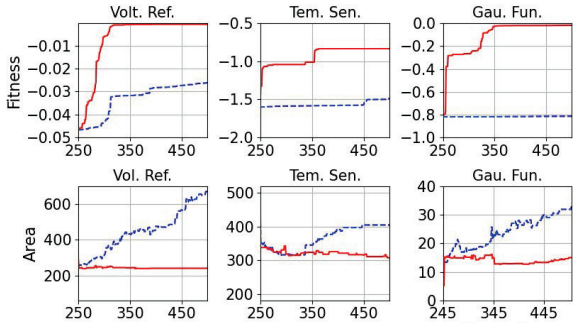

Figure 2: Average fitness results and circuit area comparison with (red solid line) and without (blue dashed line) Shapley.

Table 1: Comparisons with previous work for the three benchmark circuits

Parameters	[4]	[7]	Pre-250 Gen. (Without Shap.)	Post-250 Gen. (With Shap.)
Reference voltage				
Evaluations	5.12×10^7	5.6×10^6	2.5×10^4	$\mathbf{5{\times}10^4}$
$\|MBF\|$	6.6	2.64	0.112	**0.0054**
#Components	67	70.2	32	**15**
Area	-	-	$293.31\mu^2m$	$\mathbf{240.27}\mu^2m$
Temp. sensor				
Evaluations	1.6×10^7	6.5×10^6	2.5×10^4	$\mathbf{5{\times}10^4}$
$\|MBF\|$	26.4	1.13	0.065	0.0519
#Components	54	27.8	22	**19**
Area	-	-	$329.24\mu^2m$	$\mathbf{307.78}\mu^2m$
Gau. function				
Evaluations	2.3×10^7	4.3×10^6	2.5×10^4	$\mathbf{5{\times}10^4}$
$\|MBF\|$	0.094	0.3	0.036	**0.0147**
#Components	14	36	24	25
Area	-	-	$14.98\mu^2m$	$\mathbf{15.01}\mu^2m$

Moreover, we also compared with our proposed method with other previous evolutionary analog circuit design work [4, 7] on three benchmarks, where the comparisons are shown in Table 1. In terms of the fitness results and the number of evaluations, our proposed work achieves better fitness with less number of evaluations.

3.2 Shapley tree-based circuit evolution leads to more compact circuits

Besides the fitness results during the evolution, we are also concerned with the size of the evolved circuits. To analyze if our Shapley-oriented stage is beneficial to evolve more parsimonious circuits, we also monitor the circuit area of the individual with the best fitness in each generation. As we can see from Table 1, the circuit area of the circuit evolved with the Shapley-oriented stage is smaller than that of without Shapley-oriented stage. The visualized curve of the circuit area changed during the evolution is shown in Fig. 2 (bottom row). We take the voltage reference circuit as an example. As for the evolved results with Shapley-oriented stage, during the generations from 250 to 350, the fitness is increasing while its corresponding circuit area has no obvious increase, and during the generation from 350 to 500, even if the fitness has not witnessed an obvious increase, the area still did not increase, which is different from the blue dash line. For the evolved results without the Shapley-oriented stage, although the fitness is increasing gradually during the generation from 250 to 500, the corresponding area is growing dramatically.

The circuit area is also compared with that of the previous evolutionary circuit design work [4] [7], which is shown in Table 1. Although previous work did not give the circuit area of their evolved circuits, the number of components they have applied is accessible. Except for the Gaussian generator task, our work outperforms the existing works over other two benchmarks in terms of the the number of components.

4 CONCLUSION

In this work, we proposed a novel evolutionary framework to evolve analog circuits. The framework introduces a new strategy based on Shapley values to identify the contribution of each function node in a circuit-plausible way and guide the evolutionary process towards more promising regions of the search space. Experiments based on three benchmarks verified that our proposed framework outperformed existing evolutionary circuit designs in terms of both fitness and circuit area.

In the future, the scalability of our approach could be improved further to evolve much larger and more complex problems. As for the perspective of memristive applications, more practical factors of evolving memristive circuits will be considered in the future.

ACKNOWLEDGMENTS

This work was support by the Research Institute of Trustworthy Autonomous Systems (RITAS), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531).

REFERENCES

[1] Federico Castejón and Enrique J Carmona. 2018. Automatic design of analog electronic circuits using grammatical evolution. *Appl. Soft Comput.* 62 (2018), 1003–1018.
[2] Alan E Gelfand, Dipak K Dey, and Hong Chang. 1992. *Model determination using predictive distributions with implementation via sampling-based methods.* Technical Report. Stanford Univ CA Dept of Statistics.
[3] Amirata Ghorbani and James Zou. 2020. Neuron shapley: Discovering the responsible neurons. *arXiv preprint arXiv:2002.09815* (2020).
[4] John R Koza, David Andre, Martin A Keane, and Forrest H Bennett III. 1999. *Genetic programming III: Darwinian invention and problem solving.* Vol. 3. Morgan Kaufmann.
[5] John R. Koza, Forrest H Bennett, David Andre, Martin A. Keane, and Frank Dunlap. 1997. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Trans. Evol. Comput.* 1, 2 (1997), 109–128.
[6] Bo Liu, Yan Wang, Zhiping Yu, Leibo Liu, Miao Li, Zheng Wang, Jing Lu, and Francisco V Fernández. 2009. Analog circuit optimization system based on hybrid evolutionary algorithms. *Integration* 42, 2 (2009), 137–148.
[7] Claudio Mattiussi and Dario Floreano. 2007. Analog genetic encoding for the evolution of circuits and networks. *IEEE Trans. Evol. Comput.* 11, 5 (2007), 596–607.
[8] Alvin E Roth. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley.* Cambridge University Press.
[9] Lloyd S Shapley. 2016. *A value for n-person games.* Princeton University Press.
[10] Xinming Shi, Leandro L. Minku, and Xin Yao. 2022. A Novel Tree-based Representation for Evolving Analog Circuits and Its Application to Memristor-Based Pulse Generation Circuit. *Under review by Genetic Programming and Evolvable Machines* (2022).