Tree-based Genetic Programming for Evolutionary Analog Circuit with Approximate Shapley Value

Xinming Shi[1] [0000−0002−2053−6924], Leandro L. Minku[2] [0000−0002−2639−0671], and Xin Yao[2,3] [0000−0001−8837−4442]

[1] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK
[2] School of Computer Science, University of Birmingham, Birmingham, UK
[3] School of Data Science, Lingnan University, Hong Kong SAR
x.shi@qub.ac.uk, l.l.minku@bham.ac.uk, xinyao@ln.edu.hk

Abstract. The automated design of analog circuits presents a significant challenge due to the complexity of circuit topology and parameter selection. Traditional evolutionary algorithms, such as Genetic Programming (GP), have shown potential in this domain but are often hindered by inefficient search processes and the large design space. Furthermore, fitness evaluation in the evolutionary design of circuits is often computationally very expensive. In this paper, we introduce a novel evolutionary framework that leverages approximate Shapley values to guide the optimization process in tree-based genetic programming for analog circuit design. Our approach addresses the computational challenges associated with computing Shapley values by introducing a two-stage evolutionary framework that includes a Shapley Value Library (SV_{lib}) and a KNN-based prediction for efficient estimation of Shapley values. Our proposed work not only enhances the search efficiency by focusing on the most beneficial sub-circuits but also leads to more compact and efficient circuit designs. Furthermore, fitness evaluation in the evolutionary design of circuits is often computationally very expensive experiments, we verify that our framework accelerates evolutionary convergence and outperforms traditional methods in terms of circuit optimization.

Keywords: Tree-based genetic programming · Evovlable hardware · Shapley Value · KNN · Analog circuit design.

1 Introduction

The design of analog circuits is a critical yet challenging task in electronics, where achieving optimal configurations is essential for performance and efficiency. Traditional methods often struggle with the complexity of analog circuit design [17], making evolutionary algorithms, particularly Genetic Programming (GP) [13], a promising alternative for automating this process. However, the efficiency of these algorithms is hampered by the vast search space and the intricate interplay between circuit topology and parameters.

In evolutionary analog circuit design, genetic operators like crossover and mutation play a pivotal role, yet their efficacy is often constrained by the choice of circuit representation. Typically, these operators are applied to genes selected at random, a strategy that can inadvertently discard valuable sub-circuits, thereby diminishing search efficiency. Additionally, the evolution process might be plagued by the bloat phenomenon [14], where circuits become unnecessarily large due to components that contribute nothing to the overall functionality.

The crux of enhancing search efficiency lies in the ability to discern which parts of a sub-circuit are truly instrumental in driving evolutionary progress. Current methods such as the Leave-One-Out (LOO) approach [7], while useful in other contexts, fall short in circuit design as they fail to account for the intricate interdependencies among circuit elements. Consequently, there is a pressing need for a more apt metric that can accurately assess the significance of each gene within the circuit's framework, thereby refining the evolutionary process. In recent years, Shapley values, derived from cooperative game theory, have gained prominence as a robust tool for interpreting machine learning models, especially tree-based models. By attributing quantified contributions to each feature, SHAP values offer a transparent and consistent approach to model interpretation. Despite their theoretical appeal, the computation of SHAP values is notoriously resource-intensive, posing a significant challenge for large datasets or complex models such as deep tree structures [26]. This issue is particularly acute in evolutionary analog circuit design, where efficiently assessing the contribution of circuit components is crucial for guiding the evolutionary process toward optimal designs.

Motivated by the computational challenges associated with SHAP values and the need for a more efficient method in the context of analog circuit design, we propose an evolutionary framework that leverages approximate Shapley values to guide the optimization process. Our approach aims to enhance the search efficiency and circuit quality by retaining and exploiting beneficial sub-circuits, thereby addressing the limitations of traditional genetic operators that often operate randomly and may discard useful circuit components.

Our key contributions are as follows:

1. **Shapley Value Library Creation for Circuit Trees:** We establish a novel methodology for the computation of Shapley values in circuit tree individuals, laying the groundwork for a Shapley Value Library (SV_{lib}). This library represents a comprehensive collection of real Shapley values for nodes in circuit trees, providing a crucial reference for evolutionary operations.
2. **Two-Stage Evolutionary Framework with Accelerated Computation:** We introduce a two-stage evolutionary framework that leverages the SV_{lib}. The first stage involves the creation of this library by calculating the real Shapley values of nodes in circuit tree individuals. In the second stage, we utilize KNN-based prediction to rapidly estimate the Shapley values of nodes in new individuals. This accelerated computation significantly enhances the efficiency of the evolutionary process.
3. **Guided Evolution of Circuit Trees Using Approximated Shapley Values:** Within our evolutionary framework, we employ the predicted Shapley values to guide the crossover and mutation processes. This approach ensures that evolu-

tionary operations are informed by a node's importance, directing the evolution of analog circuits towards more promising regions of the search space.

4. **Experimental Studies to Show Enhanced Evolutionary Efficiency and Circuit Optimization:** Experiments shows that our approach accelerates evolutionary convergence and produces more efficient circuit designs. This verifies the effectiveness of integrating Shapley value computation with tree-based genetic programming in circuit evolution.

The rest of this paper is structured as follows. Section 2 introduces the related work. Section 3 proposes a novel genetic programming approach for evolving analog circuits. Section 4 describes the experimental studies for verifying our proposed method. The conclusions of our work are presented in Section 5.

2 Related Work

2.1 Preliminary Knowledge of Shapley Value

In Cooperative Game Theory (CGT) [3], a set of N players are interconnected through a score function $V : 2^N \to \mathbb{R}$, where $V(S)$ represents the performance of the model after setting the elements in $N \setminus S$ to zero. To distribute the collective reward among the players equitably, the Shapley value [25] is introduced. It quantifies the contribution of player i to the coalition by defining the marginal contribution $\Delta_V(i, S)$ as the additional value generated by including i in S:

$$\Delta_V(i, S) = V(S \cup i) - V(S) \tag{1}$$

The Shapley value is essentially the average of the marginal contributions across all possible subsets of players, considering the permutations where a particular ordering of S immediately precedes player i:

$$u_i = \frac{1}{|N|} \sum_{S \subseteq N \setminus i} \frac{\Delta_V(i, S)}{\binom{|N|-1}{|S|}} \tag{2}$$

This formulation accounts for interactions between players, capturing scenarios where the performance improvement is contingent on the presence or absence of specific players.

Shapley values have found applications beyond traditional game theory, such as in feature attributions for machine learning models, where they offer insights consistent with human intuition [19]. They have also been used to evaluate the importance of training samples [12, 8] and to assess the contribution of individual elements within a model, such as neurons in a neural network [28]. These applications underscore the versatility and relevance of Shapley values in various domains.

2.2 Evolutionary Design of Analog Circuits

In the realm of automated circuit design, evolutionary algorithms have emerged as a powerful tool, with a plethora of approaches being explored [16, 13, 4, 10]. One notable example is the work of Kruiskamp et al. [15], who leveraged Genetic Algorithms (GA) to tackle the synthesis of CMOS operational amplifiers (opamps). In their approach, each individual in the population represented a potential circuit design encoded as a multi-gene chromosome, which could be decoded into an actual circuit. In a different way, Grimbleby et al. [9] utilized GA for the automated synthesis of analog networks, focusing on configuring the circuit structure. However, this approach necessitated numerical optimization to ascertain the values of the circuit components, adding a layer of complexity to the design process.

Moreover, approaches based on Genetic Programming (GP) have shown the ability to evolve circuit netlists that encompass both topology and device values, offering a more integrated solution. While some researchers have used GA to encode circuit topology and parameter values as strings, this often results in limited circuit diversity and a cumbersome decoding process [18, 6]. In contrast, GP-based methods enable a richer variety of circuit designs and a more streamlined decoding process, making them a promising avenue for advancing the field of analog circuit design.

2.3 Knowledge-driven Evolutionary Operators

In standard Genetic Programming (GP), crossover and mutation operators play a crucial role in generating offspring. However, the random selection of genes for these evolutionary operations may hinder search efficiency by overlooking potentially valuable sub-circuits [11]. Additionally, the presence of devices with zero contribution can lead to bloat, resulting in larger evolved circuits [11]. To overcome these limitations, researchers have explored more sophisticated approaches that incorporate semantic information to enhance the exploration of the search space [5, 21, 24]. For instance, Beadle et al. [2] utilized semantic information to guide GP crossover in Boolean problem domains, while Krawiec et al. [15] defined the semantics of an individual as a vector of outputs for corresponding input fitness cases. Nguyen et al. [23] proposed a semantic crossover approach for real-value domains. However, these advanced operations are tailored for Boolean or real-value problem domains and are not directly applicable to analog circuit design, where circuit validity is a crucial consideration [30].

In the context of evolutionary analog circuit design, it is essential to account for the dependencies among sub-circuits and assign appropriate importance measures. Moreover, the concept of equivalent circuits, which refers to circuits with identical input-output characteristics as the original circuit [1], should be integrated into the design process. While there has been an attempt to evaluate the importance of sub-circuits using the LOO strategy [11], this approach neglects the dependencies among different sub-circuits/devices and their combinations, and it has been limited to digital circuits. Analogue circuits handle continuous signals, while digital circuits use binary signals. To the best of our knowledge, there is no existing work in evolutionary analog circuit design that comprehensively addresses these two perspectives. As discussed in Section 2.1, the Shapley value offers a promising solution for measuring the importance of sub-circuits/devices while considering their dependencies.

3 Our Method

We introduce a novel genetic programming approach for evolving analog circuits, utilizing an approximated Shapley Value for enhanced efficiency.

The algorithm commences with a population of circuit trees, a Shapley Value Library, and an archive for individual records. It operates in two stages: initially, it calculates real Shapley values for nodes in each circuit tree, updating the library and archive. Beyond a certain threshold, it employs KNN to predict Shapley values using historical data, guiding the genetic operations of crossover and mutation. This process iterates until a set number of iterations are reached, yielding a refined population of circuit trees. This method innovatively integrates Shapley values into circuit evolution, streamlining the search process and improving design outcomes. The specific description of our proposed approach is presented as Algorithm 1.

In this section, we introduce the overarching goal of the algorithm: to efficiently compute the Shapley values of nodes within circuit tree individuals, which represent potential solutions to a given circuit design problem. The initialization process is crucial as it sets up the initial population of circuit tree individuals, denoted as P_t Each individual is a tree-like structure where nodes represent different circuit components, and the connections between nodes define the circuit's topology.

Additionally, the Shapley Value Library (SV_{lib}) is established. This library is a vital component of the algorithm, as it stores the Shapley values of nodes, providing a measure of their importance or contribution to the overall performance of the circuit. An archive is also set up to store individuals alongside their computed Shapley values, creating a historical record that will be instrumental in predictive modeling during later stages of the algorithm.

3.1 Tree-based Hierarchical Circuit Encoding Method

In evolutionary analog circuit design, it is crucial to consider both the evolution of circuit topology and the optimization of device values. Our approach represents circuits using a multi-tree structure, T, where the set of all nodes in the tree is denoted as M_T. The internal nodes, or function nodes, are represented by the set N_F. Each function node consists of two parts: the device type and a value tree. The value tree is a binary tree that represents the numeric value of the circuit device, with internal nodes for arithmetic operations and leaf nodes for numeric values. For devices that do not require a value, the value tree is null.

The leaf nodes, or terminal nodes, are denoted by the set N_T. Each terminal node represents the position of one device port in the circuit netlist. The arity of a function node, which is the number of child nodes, is determined by the number of device ports. We define $Ch_k(node_i)$ as the k-th child of function node $node_i$. To enhance flexibility in circuit representation, devices with polarity are represented by different function nodes, allowing for any-connection circuits.

To transform the tree-based circuit representation into circuit netlists, each function node is assigned a netlist position number, determined by the left-most terminal nodes corresponding to all its children. For example, consider a circuit with function nodes MOS, R, C, and Mem, and terminal nodes $\{1, 2, 3, 4, 5, 6\}$. The hierarchy

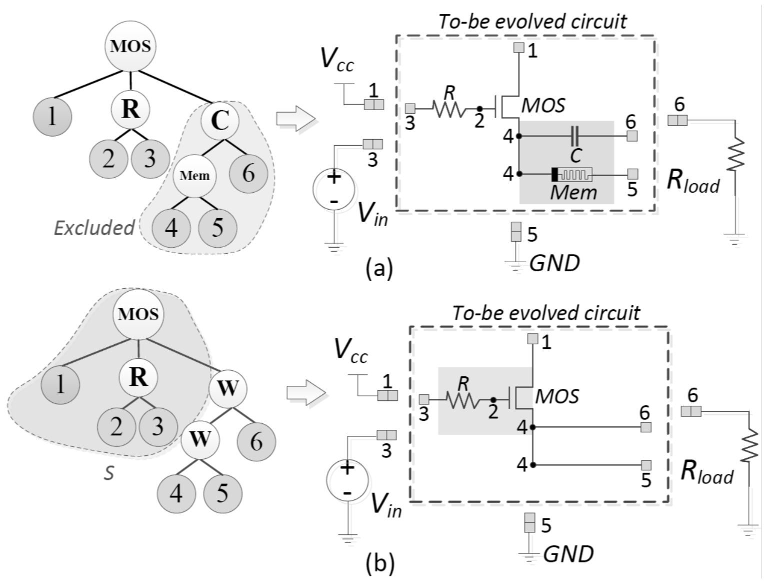

Fig. 1. (a) An example of tree encoding method for a circuit. (b) Equivalent sub-circuit during Shaley value calculation.

formed by these nodes defines the circuit connection, with the position of each node given by a function $U(H(node_i))$, where $H(node_i)$ is defined based on whether the node is a terminal or function node.

To ensure circuit feasibility, three strategies are applied: avoiding dangling terminals in the embryo circuit, preventing dangling terminals in the evolved circuit, and restricting tree depth. An embryo circuit is a basic initial circuit that needs to be connected to the evolved circuit to form a complete circuit loop. The evolved circuit must ensure that each terminal node is used at least twice to prevent dangling terminals. Finally, a maximum depth limit is imposed on the tree to prevent tree bloat, with any nodes exceeding this limit replaced by their left terminal node. More details of tree-based hierarchy circuit encoding and decoding methods could be found in work [27].

3.2 Shapley Value-based Evaluation of Sub-circuits to Guide Genetic Operators

In order to evaluate the contribution of the function node to the whole circuit tree representation, several desirable proprieties to evaluate the circuit device should be taken into the consideration. We list these properties below:

- **Zero contribution:** One decision to make is how to handle circuit devices/blocks that have no contribution. We say that a function node i has no contribution if $\forall S \subseteq N \setminus \{i\} : V(S \cup \{i\}) = V(S)$. This means that it does not change the performance when added to any subtree in of the whole tree. For such null function node, the valuation should be 0.

- **Symmetric elements:** If two nodes contribute exactly the same to any subset of the rest of function nodes, they will have the same values by definition. Mathematically, if $\forall S \subseteq N \setminus \{i\} : V(S \cup \{i\}) = 0$.
- **Additivity in Performance Metric:** As for a circuit device and circuit evolution task, there are two or more performance metrics $V_1, V_2, ...$ for an evolved circuit. For example V_1 measures it's performance on output accuracy and V_2 is its performance on circuit area. A natural way to measure the overall performance of the model is having a linear combination of such metrics $e.g: V = V_1 + V_2$. The additivity in our context is an optional property, since the circuit performance of output accuracy is more dominated compared with other metrics due to its critical impact on reliability and effectiveness in applications.

The Shapley Value-based importance u_{node_i} of a function node $node_i$ can be calculated using the following formula, which uniquely satisfies all these properties:

$$u_{node_i} = \frac{1}{|N_F|} \sum_{S \subseteq N_F - \{i\}} \frac{V(SubTree_{S \cup node_i}) - V(SubTree_S)}{\binom{|N_F|-1}{|S|}} \tag{3}$$

where $V(S)$ denotes the performance of subtree S. To evaluate the fitness $V(s)$ of subtrees, we consider two scenarios:

(1) If the subtree π_j has overlapping nodes with the subtrees $\pi_i \in \pi, i < j$, the fitness is calculated based on the differences between the voltages on the input and output terminals of the subtree and the target voltage.
(2) If the subtree π_j has no overlapping nodes with the subtrees $\pi_i \in \pi, i < j$, its fitness is calculated based on the maximum fitness of the previous subtrees or the fitness of π_j itself.

Figure 1 (b) illustrates the example of evaluating the Shapley value within the context of a circuit. To ascertain the contribution of a specific function node, it is temporarily omitted from the circuit and substituted with a distinct node, denoted by W. This node W signifies a wire connection, effectively short-circuiting the circuit element whose contribution is under evaluation. By comparing the circuit's performance, before and after this substitution, we can discern the impact of the excluded node, thereby quantifying its individual contribution to the overall circuit functionality. This process enables a precise calculation of each element's Shapley value based on the variance in fitness it induces.

This Shapley Value-based approach provides an equitable assignment of values to nodes, enabling the evaluation of the contribution of sub-circuits in a circuit-plausible way. As in [26], crossover will swap the subtree rooted by the function node whose Shapley value is the highest in the one parent with the one whose Shapley value is the lowest in the other parent. Mutation will delete or replace the function node with the lowest Shappley value by a new randomly generated one.

3.3 Two-stage based Evolutionary Framework

In the initial stage, the focus is on populating the Shapley Value Library SV_{lib} with real Shapley values. For each individual i the population P_t, the algorithm computes the real Shapley value for each node, reflecting its contribution to the individual's overall performance. These calculated Shapley values are then used to update SV_{lib}, ensuring that it contains the most recent and accurate information. Concurrently, the individual i and its node Shapley values are stored in the archive, providing a rich dataset for future predictions.

The second stage of the algorithm highlights our novel methodology. As the iteration count t surpasses the predefined threshold T_1, the algorithm transitions to using a KNN (K-Nearest Neighbors) based approach for predicting the Shapley values of nodes in new individuals.

3.4 KNN-Based Approximation of Shapley Values

Once the iteration count exceeds the predefined threshold T_1, our proposed approach uses KNN to predict the Shapley values of nodes in new individuals, leveraging historical data from the archive for quick estimation of node importance. The predicted values then guide genetic operations such as crossover and mutation, instead of the Shappley values themselves. This enables the algorithm to focus on nodes with higher importance to create offspring with improved performance, while avoiding the computational cost of computing the Shapley values. After these operations, the population P_t is updated with the new, evaluated individuals.

The integration of KNN-based predictions with genetic operations is a key feature, enabling efficient exploration of the solution space. As the algorithm reaches the maximum number of iterations ($maxiter$), it concludes with a final refined population P_t. The innovative use of KNN for predicting Shapley values, combined with strategic genetic operations, makes the algorithm a powerful tool for circuit design optimization.

The algorithm effectively integrates the concept of Shapley values into the evolutionary process of circuit trees, utilizing a two-stage approach to enhance efficiency. The first stage is dedicated to building a comprehensive Shapley Value Library, while the second stage accelerates the computation by predicting Shapley values using KNN, thereby guiding the evolutionary operations more effectively.

4 Experimental Studies

The main loop of our evolutionary framework is developed in Python, while the performance evaluation of the circuits is conducted through simulations in NGSPICE [29], a tool derived from Spice3 [22].

The experimental study encompasses three distinct types of circuit evolution tasks, focusing on the evaluation of the proposed method across three circuits: a voltage reference circuit, a temperature sensor circuit, and a Gaussian function generator. These circuits are commonly utilized in assessing the efficacy of evolutionary analog circuit design methods and are referenced extensively in the literature [14, 20, 4], demonstrating their relevance and applicability to this field. Parameter setting is given in Table 1.

Reference Voltage Circuit In this task, the objective is to design and refine a reference voltage circuit that consistently delivers a fixed output of $2V$. To assess the

Algorithm 1 Shapley Value Approximation Using KNN for Circuit Tree Evolution

```
1: Input: Circuit tree individuals, target
2: Output: Individuals after crossover and mutation based on importance
3: Initialize population $P_t$, Shapley Value Library $SV_{lib}$, and archive $Archive$
4: while $t < maxiter$ do        ▷ Iterate until max number of iterations is reached
5:     if $t < T_1$ then        ▷ First Stage: Shapley Value Library Creation
6:         for each individual $i$ in $P_t$ do
7:             Calculate real Shapley value of each node in $i$
8:             Update $SV_{lib}$ with real Shapley values
9:             Store individual $i$ and its node Shapley values in $Archive$
10:        end for
11:    else        ▷ Second Stage: Accelerated Computation with KNN Prediction
12:        Use KNN to predict Shapley values of nodes in new individuals using $Archive$
13:        for each new individual $i$ in $P_t$ do
14:            Predict importance of nodes in $i$ using KNN with $Archive$
15:            Perform crossover and mutation on $i$ guided by predicted importance
16:        end for
17:        Evaluate new individuals and integrate into the population
18:    end if
19: end while
20: Return $P_t$
```

Table 1. Parameter setting

Algorithm Parameter	Value
Population Size (Pop_Size)	100
Tournament Size (Tou_Size)	20
Maximum Iterations (Max_Iteration)	500
Crossover Probability (P_{cross})	0.8
Mutation Probability (P_{mutation})	0.2
K Value (K)	10
Threshold (T_1)	50
Crossover Value Probability ($P_{\text{valuecross}}$)	0.2

circuit's performance across a range of temperatures, the output voltage is measured at various points for each temperature condition. These measurements are compared against predetermined ideal values to evaluate circuit accuracy. The effectiveness of a given design is quantified by a fitness function, which is a summation of the squared deviations between the measured and target voltages, adjusted for a margin of error. Only deviations exceeding a threshold of $0.01V$ are considered, reflecting the precision goal of the circuit design [14]. The embryo circuit setting of reference voltage circuit is presented in our previous work [27]. The fitness function is given as the following equation [14]:

$$fitness = -\sum_{i,j} \varepsilon_{ij}, \tag{4}$$

where ε_{ij} is :

$$\varepsilon_{ij} = \begin{cases} \left(V_{outi,j} - V^*_{outi,j}\right)^2, \text{if } |V_{outi,j} - V^*_{outi,j}| \geq 0.01V \\ 0, \qquad\qquad \text{if } |V_{outi,j} - V^*_{outi,j}| < 0.01V. \end{cases} \tag{5}$$

At various circuit temperatures T_i, each measured output voltage point $V_{outi,j}$ corresponds to a target value $V^*_{outi,j}$. i represents the i-th circuit temperature, and j represents the sampled output voltage points.

Temperature Sensing Circuit The challenge involves developing a circuit capable of sensing temperature changes, as reflected by variations in its output voltage. The output voltage linearly correlates with temperature. The linear relationship is characterized by a constant, ensuring output voltage directly corresponds to ambient temperature changes. The circuit's performance is evaluated through a fitness function, which aggregates the squared differences between actual and expected output voltages. This approach allows for precise calibration of the circuit's temperature sensitivity, leveraging genetic algorithms to fine-tune its response characteristics. The embryo circuit setting of temperature sensor is presented in our previous work [27]. The fitness function is defined as following [14]:

$$fitness = -\sum_i \left(V_{outi} - V^*_{outi}\right)^2. \tag{6}$$

At various circuit temperatures T_i, the output voltage V_{out} changes and will be measured for evaluation. The target value of the output voltage for the i-th temperature, V^*_{outi}, is linearly related to T_i and is defined as $V^*_{outi} = \eta T_i$. Here, η is a constant representing the linear relationship between the circuit temperature and the output voltage.

Gaussian Function Generator The task focuses on the creation of a Gaussian function generator, where the aim is to produce an output current that fits a Gaussian distribution in relation to the input voltage. The task is to measure the output current for various input voltages and align these measurements with their theoretical Gaussian counterparts. The alignment is measured using a fitness function with a key normalization factor. This factor adjusts for the scale of the current measurements, ensuring the squared differences between expected and actual currents are accurately compared. This process embodies the application of evolutionary algorithms to the intricate task of circuit function generation, following the methodologies proposed by [14]. The embryo circuit setting of Gaussian function generator is given in our previous work [27]. The fitness function is defined as follows [14]:

$$fitness = -10^{14} * \sum_i \left(I_{outi} - I^*_{outi}\right)^2. \tag{7}$$

During circuit evolution, the output current is measured for evaluation. Different input voltages V_{in1} will correspond to different target values of the output current.

Table 2. Average fitness under ablation experiments

Tasks	Cases	|BF|	|MBF|
Voltage reference circuit	Random	0.0051	0.0261
	TMC SV	0.0027	0.0183
	KNN SV	0.0012	0.0142
Temperature sensor	Random	0.0189	0.3981
	TMC SV	0.0096	0.1194
	KNN SV	0.0087	0.1048
Gaussian function generator	Random	0.0874	0.4046
	TMC SV	0.0382	0.1988
	KNN SV	0.0396	0.1208

4.1 Ablation Study

Table 4.1 gives the average fitness under ablation experiments, focusing on the performance of three approaches (*Random*, *TMC SV*, and *KNN SV*) across three different tasks: a voltage reference circuit, a temperature sensor, and a Gaussian function generator. This approach helps identify which parts are essential and how each component influences the overall effectiveness. Random approach refers to the approach where random nodes are selected for crossover and mutation. The *Random* method serves as a baseline, wherein mutation rates, crossover points, and selection mechanisms are randomized, allowing for straightforward comparisons with more sophisticated strategies. *TMC SV* refers to the approach where the Truncated Monte Carlo Shapley Value (*TMC SV*) is applied. This approach simplifies the computational cost of calculating the exact Shapley values by using a truncated Monte-Carlo technique [26], maintaining a balance between computational cost and accuracy. The parameter setting is the same as the work in [26]. *KNN SV* refers to our proposed approach where the KNN model is applied to the predict the Shapley Value. Each task evaluates the best fitness (|BF|) and mean best fitness (|MBF|) achieved by each method. In our study, we employ three distinct methods for guiding genetic operations.

For the voltage reference circuit, the *KNN SV* method outperforms the others, showing the lowest best fitness and mean fitness, indicating a superior capability to optimize circuit parameters effectively. In the temperature sensor task, *KNN SV* again demonstrates its efficacy with the lowest best fitness and a competitive mean fitness, suggesting its robustness and reliability in sensor optimization. Lastly, for the Gaussian function generator, while *KNN SV* does not achieve the lowest best fitness, it offers a significantly lower mean fitness compared to *TMC SV*, highlighting its consistency and effectiveness in generating functions with high fidelity.

Overall, the *KNN SV* method consistently shows promising results across all tasks, proving its potential as a highly effective tool in these specific applications. Its ability to consistently achieve low best and mean fitness values suggests it might be the preferred method for similar tasks, though specific requirements and goals of each experiment should guide the final methodology choice.

Table 3. Comparisons with previous works

Parameters	[14]	[20]	[27]	Ours
Reference voltage				
Evaluations	5.12×10^7	5.6×10^6	5×10^4	$\mathbf{5{\times}10^4}$
$\lvert MBF \rvert$	6.6	2.64	0.0261	**0.0142**
#Components	67	70.2	32	**15**
Temperature sensor				
Evaluations	1.6×10^7	6.5×10^6	5×10^4	$\mathbf{5{\times}10^4}$
$\lvert MBF \rvert$	26.4	1.13	0.3981	0.1048
#Components	54	27.8	22	**19**
Gaussian function				
Evaluations	2.3×10^7	4.3×10^6	5×10^4	$\mathbf{5{\times}10^4}$
$\lvert MBF \rvert$	0.094	0.3	0.4046	**0.1208**
#Components	14	36	24	25
P-value	[14] VS ours: 0.0404; [20] VS ours:0.0404; [27] VS ours: 0.0452			

4.2 Comparisons with Previous Work

Table 3 presents a comprehensive comparison between our work and three previous studies, applied to the design of reference voltage circuits, temperature sensors, and Gaussian function generators. A key focus is on the number of evaluations, mean best fitness (|MBF|), and the number of components utilized in each approach.

In the reference voltage task, our proposed approach alongside Shi [27] drastically reduces the number of evaluations to just 50,000, a significant decrease from the millions required in earlier works by Koza [14] and Mattiussi [20]. Moreover, our proposed approach achieves the lowest mean best fitness at 0.0142 and uses the fewest components (15), indicating a substantial improvement in circuit optimization efficiency and precision.

Similarly, for the temperature sensor application, both the recent study and Shi [27] have again significantly cut down the evaluation count. Our work excels with a mean best fitness of 0.1048, which is the best among all compared studies, and achieves this with fewer components (19), demonstrating an optimized and efficient design.

The Gaussian function generator results mirror these improvements, with all recent studies requiring fewer evaluations. Although our proposed approach does not achieve the lowest historical mean best fitness, it performs significantly better than Shi [27] with a fitness of 0.1208 and uses a moderate number of components (25), balancing complexity and performance efficiency.

Statistically significant improvements in the current methods over previous studies are confirmed by P-values of 0.0404 and 0.0452. These values indicate a significant enhancement in performance across all metrics, validating the effectiveness of the new approaches.

5 Conclusion

In this paper, we focus on the significant challenges of automated analog circuit design due to the complexity of circuit topology and parameter selection. While traditional evolutionary algorithms like Genetic Programming (GP) have shown potential in this field, they often struggle with inefficient search processes and the vast design space. To address these issues, we introduce a novel evolutionary framework that uses approximate Shapley values to guide the optimization process in tree-based genetic programming for analog circuit design. Our approach reduces the computational costs of Shapley values by implementing a two-stage evolutionary framework. This includes the creation of a Shapley Value Library (SV_{lib}) and a KNN-based prediction phase for quickly estimating Shapley values. Our approach improves search efficiency by focusing on the most beneficial sub-circuits, leading to more compact and efficient circuit designs. Through experimental verification, we demonstrate that our approach accelerates evolutionary convergence and surpasses traditional methods of evolving circuits. Our future includes the scalability of our approach to larger analog circuits.

References

1. Allen, P.E., Holberg, D.R.: CMOS analog circuit design. Elsevier (2011)
2. Beadle, L., Johnson, C.G.: Semantically driven crossover in genetic programming. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). pp. 111–116. IEEE (2008)
3. Branzei, R., Dimitrov, D., Tijs, S.: Models in cooperative game theory, vol. 556. Springer Science & Business Media (2008)
4. Castejón, F., Carmona, E.J.: Automatic design of analog electronic circuits using grammatical evolution. Appl. Soft Comput. **62**, 1003–1018 (2018)
5. Ffrancon, R., Schoenauer, M.: Memetic semantic genetic programming. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation. pp. 1023–1030 (2015)
6. Gan, Z., Yang, Z., Shang, T., Yu, T., Jiang, M.: Automated synthesis of passive analog filters using graph representation. Expert Syst. Appl. **37**(3), 1887–1898 (2010)
7. Gelfand, A.E., Dey, D.K., Chang, H.: Model determination using predictive distributions with implementation via sampling-based methods. Tech. rep., Stanford Univ CA Dept of Statistics (1992)
8. Ghorbani, A., Zou, J.: Data shapley: Equitable valuation of data for machine learning. In: International Conference on Machine Learning. pp. 2242–2251. PMLR (2019)
9. Grimbleby, J.B.: Automatic analogue network synthesis using genetic algorithms. In: Proc. GALESIA 1995. pp. 53–58 (Sheffield, 1995)
10. He, J., Yin, J.: Evolutionary design model of passive filter circuit for practical application. Genetic Programming and Evolvable Machines **21**(4), 571–604 (2020)
11. Hodan, D., Mrazek, V., Vasicek, Z.: Semantically-oriented mutation operator in cartesian genetic programming for evolutionary circuit design. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 940–948 (2020)
12. Jia, R., Dao, D., Wang, B., Hubis, F.A., Hynes, N., Gürel, N.M., Li, B., Zhang, C., Song, D., Spanos, C.J.: Towards efficient data valuation based on the shapley value. In: The 22nd International Conference on Artificial Intelligence and Statistics. pp. 1167–1176. PMLR (2019)
13. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.A.: Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming. In: Proc. 1st Annual Conference on Genetic Programming. pp. 132–140 (Stanford, 1996)
14. Koza, J.R., Andre, D., Keane, M.A., Bennett III, F.H.: Genetic programming III: Darwinian invention and problem solving, vol. 3. Morgan Kaufmann (1999)

15. Krawiec, K., Wieloch, B.: Functional modularity for genetic programming. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 995–1002 (2009)
16. Kruiskamp, W., Leenaerts, D.: Darwin: Cmos opamp synthesis by means of a genetic algorithm. In: Proc. 32nd DAC 1995. pp. 433–438 (San Francisco, 1995)
17. Liu, B., Wang, Y., Yu, Z., Liu, L., Li, M., Wang, Z., Lu, J., Fernández, F.V.: Analog circuit optimization system based on hybrid evolutionary algorithms. Integration **42**(2), 137–148 (2009)
18. Lohn, J.D., Colombano, S.P.: A circuit representation technique for automated circuit design. IEEE Trans. Evol. Comput. **3**(3), 205–219 (1999)
19. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st international conference on neural information processing systems. pp. 4768–4777 (2017)
20. Mattiussi, C., Floreano, D.: Analog genetic encoding for the evolution of circuits and networks. IEEE Trans. Evol. Comput. **11**(5), 596–607 (2007)
21. Moraglio, A., Krawiec, K.: Geometric semantic genetic programming for recursive boolean programs. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 993–1000 (2017)
22. Nagel, L., Pederson, D.O.: Spice (simulation program with integrated circuit emphasis) (1973)
23. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantic aware crossover for genetic programming: the case for real-valued function regression. In: Genetic Programming: 12th European Conference, EuroGP 2009 Tübingen, Germany, April 15-17, 2009 Proceedings 12. pp. 292–302. Springer (2009)
24. Pawlak, T.P., Krawiec, K.: Competent geometric semantic genetic programming for symbolic regression and boolean function synthesis. Evolutionary computation **26**(2), 177–212 (2018)
25. Roth, A.E.: The Shapley value: essays in honor of Lloyd S. Shapley. Cambridge University Press (1988)
26. Shi, X., Gao, J., Minku, L.L., Yao, X.: Evolving parsimonious circuits through shapley value-based genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 602–605 (2022)
27. Shi, X., Minku, L.L., Yao, X.: A novel tree-based representation for evolving analog circuits and its application to memristor-based pulse generation circuit. Genetic Programming and Evolvable Machines **23**(4), 453–493 (2022)
28. Stier, J., Gianini, G., Granitzer, M., Ziegler, K.: Analysing neural network topologies: a game theoretic approach. Procedia Computer Science **126**, 234–243 (2018)
29. Vogt, H., Hendrix, M., Nenzi, P.: Ngspice user's manual version 31 (describes ngspice release version) (2019)
30. Zhao, Z., Zhang, L.: An automated topology synthesis framework for analog integrated circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(12), 4325–4337 (2020)