



Automated Digital Filter Synthesis with Canonical Signed Digit via Genetic Algorithms

Simone Deiana, Luca Manzoni, Andrea De Lorenzo, Luigi Rovito

Department of Engineering and Architecture, University of Trieste, Italy

*E-mails: simone.deiana@studenti.units.it lmanzoni@units.it andrea.delorenzo@units.it
luigi.rovito@units.it*

Abstract: *Finite Impulse Response (FIR) digital filters are widely used in signal processing to remove noise and extract frequency-domain features. Genetic Algorithms (GAs) have been employed to synthesize FIR filters by optimizing the coefficients in their transfer function. A notable approach is to evolve FIR filter coefficients in the Canonical Signed Digit (CSD) representation, which simplifies hardware implementation by replacing multipliers with shifts and additions. However, its crossover and mutation operators frequently produce offspring that violate CSD constraints, requiring repeated retries or coefficient resets. We propose modified crossover and mutation operators that guarantee CSD-compliant offspring by construction. We compare our method against the original CSD-based GA and a real-valued GA baseline across five low-pass filter design problems, varying filter order and coefficient word length. Results show that our method consistently outperforms the original CSD-based approach.*

Keywords: *Evolutionary algorithms, Evolutionary computation, Digital filter, Hardware.*

1. Introduction

In signal processing, filters are employed to either remove noisy components of a signal or extract useful features (e.g., components within a predefined frequency range). These filters are broadly categorized into analog and digital filters. Analog filters use analog electronic circuits, whereas digital filters employ a digital processor to perform numerical calculations on sampled signal values. Digital filters

offer several advantages over their analog equivalents, including programmability, versatility, and ease of implementation on general-purpose computers.

An important class of digital filters is the Finite Impulse Response (FIR) filters [1], i.e., non-recursive filters defined by a vector of coefficients that specify a transfer function, whose output depends only on the current and previous inputs.

The problem of synthesizing digital filters, and FIR filters in particular [1, 2], is of great importance in signal processing since analog filters are often replaced by their digital equivalent [3], given the advantages the digital version offers. An early work focused on generating digital filters from given analog filters with least-squares fitting [3]. Another work focused on synthesizing FIR filters with a complex-valued, desired frequency response via standard linear programming [1]. Additionally, previous works have studied the number of multipliers required for implementing interpolated FIR filters [4] and extended the alternation theorem, which is at the core of efficient real Chebyshev approximation algorithms, to design FIR filters [5].

The early works, while obtaining notable results, are tied to underlying assumptions around the filter design process, such as the availability of target analog filters and the presence of complex-valued frequency ranges. Moreover, implementing digital filters using infinite-precision coefficients introduces truncation and quantization errors in finite-precision hardware. These errors hinder the filter's performance because they alter its coefficients. In this regard, even though mathematically sound, infinite precision is not feasible for hardware implementation due to cost and complexity. As a matter of fact, employing a discrete form with finite wordlength coefficients has the advantage of mitigating the aforementioned issues [2], where the Wordlength (W) is the number of bits our hardware can handle.

Genetic Algorithms (GAs) represent a class of Evolutionary Algorithms (EAs) that, inspired by the concept of natural evolution, optimize a population of candidate solutions (i.e., individuals) via biologically inspired operators (i.e., selection, crossover, mutation) to address a specific optimization and search task [6-10]. In the context of filter design, GAs represent a promising avenue of research since these evolutionary approaches and their encoding scheme have the potential to automatically infer filter coefficients in discrete form without a quantization process [11, 8, 2, 12].

GAs have already been investigated in previous works to create 1-D FIR filters using Power-Of-Two (POT) coefficients, resulting in multiplier-less digital filters [11-13]. More generally, GAs have been employed to address a wide range of different, specific tasks under different implementation settings [14]. Building upon

the idea of these works, Lee et al. [2] proposed an important extension of POT filter design by implementing a modified GA to devise digital 1-D FIR filters with coefficients guaranteed to be in Canonical Signed Digit (CSD) format. In addition, filter implementations are less complex and expensive since full-size multipliers, which have a higher cost in hardware, typically are not adopted in this structure. The solution encoding is based on representing filter coefficients as ternary strings corresponding to POT. To the best of our knowledge, the aforementioned research work represents the first concrete approach in the context of CSD-compliant 1-D digital FIR filter design [2].

One of the main drawbacks of the approach presented in [2] concerns compliance issues. The genetic operators employed, while being standard, can break the CSD constraints (adjacent non-zeros) or exceed the maximum number of non-zero digits allowed per coefficient. When these constraints are violated, the algorithm attempts to redo crossover and mutation up to a fixed number of trials. In case the constraints are still not satisfied, the coefficients of the filters are reset, leading to many individuals that are invalidated in the case of higher mutation probabilities or a greater number of allowed non-zero digits.

Successive works [15-17] tried to devise techniques within GAs to mitigate the CSD-related compliance issues of the technique presented in [2]. However, to accomplish this task and maintain CSD compliance, the aforementioned works implemented strategies that force the exploration algorithm to look only for specific, peculiar coefficient patterns, thereby limiting the discovery of several potentially better solutions. We want to point out that many of the older methods using CSD representations are, unfortunately, difficult to reproduce since they lack essential reproducibility indications like complete experiment results or complete algorithm description or implementation, leaving a significant number of arbitrary choices to any modern reimplementation.

There have also been modern investigations using non-mainstream computational intelligence algorithms [18-24]. A worth-mentioning project is represented by [25], where a hybrid GA is proposed to discover FIR filters with a local search mechanism that helps mitigate premature convergence. Other recent notable works are: [26], where a hybrid and dynamic mutation strategy is implemented within Differential Evolution (DE) to determine filter coefficients, and [27], in which a Long Short-Term Memory (LSTM) model is used to generate FIR filters, with Particle Swarm Optimization (PSO) employed to optimize the LSTM hyperparameters. The main limitation of these latter works is related to the real-valued solution representation they employ, which, as already mentioned, makes the practical HW mapping difficult and expensive.

In this paper, we build upon [2], and we extend the approach by defining and employing novel crossover and mutation operators that force CSD constraints to always be satisfied. These newly introduced genetic operators represent the main contribution of our study, since they enable us to always generate CSD-compliant offspring by construction, differently from the original approach where many invalid individuals are likely to be generated.

Our CSD-preserving crossover exchanges entire coefficients rather than breaking the structure of individual coefficients, thereby avoiding invalidating the offspring. By considering only the coefficients during mating, we guarantee that offspring are valid, provided that the parents are valid. On the other hand, our CSD-preserving mutation forces adjacent digits to zero if adjacent non-zero digits are generated during the mutation of the individual and refuses to mutate if the non-zero digits constraint is broken. This fix prevents invalid states without discarding the operation.

We test our approach on five low-pass target filters, in increasing order of sharpness (i.e., complexity). We perform a systematic comparison between our “Enhanced CSD GA” method, the original “Basic CSD GA” from [2], and an ideal real-valued GA as proof-of-concept. Moreover, we perform an ablation study on the selection operator of GA by testing both roulette wheel selection, as in the original version, and tournament selection. The results show that our approach achieves better qualitative performance than the basic CSD regardless of the selection operator employed, demonstrating that our design and implementation choices effectively improve upon the original algorithm proposed in [2] while guaranteeing that CSD constraints are satisfied.

The remainder of the paper is as follows: in Section 2, we describe the preliminary notions regarding both digital filters and GAs; in Section 3, we present our methodology; in Section 4, we present our experimental study along with the research questions; in Section 5, we provide answers to our research questions, and in Section 6 we conclude and outline possible future research directions.

2. Background

In this section, we describe the preliminary concepts regarding both digital filters and evolutionary computation, with special focus on GAs, which represent the technique adopted in this study.

2.1. Digital filters

A digital FIR filter is represented by the following transfer function:

$$(1) \quad H(z) = \sum_{n=0}^N a_n z^{-n},$$

where N is the order of the filter, and a_n is defined as the n -th filter coefficient, and z is the input. The parameter γ represents the vector of filter coefficients $[a_0, a_1, a_2, \dots, a_{n-1}, a_n]$. By determining the values of a_n , for each $n \in [0, \dots, N]$, different types of filters can be implemented, such as low-pass, high-pass, and band-pass filters [2].

The problem of digital FIR filter design consists of finding the coefficients of order N to match a desired frequency response. In this regard, an individual can be naturally represented as a vector. $x \in \mathbb{R}^{N+1}$, since a filter of order N has $N + 1$ coefficients (the first coefficient is related to a power of zero). However, representing these coefficients by using infinite-precision real numbers is not feasible in actual hardware implementations. To this end, actual, realistic representations employ hardware-friendly structures, such as binary strings or discrete-value vectors [2]. Furthermore, in hardware, multipliers are more expensive than adders and shifters, and hardware costs increase with the filter order and coefficient precision. Therefore, there is a need for an encoding scheme more sophisticated than a trivial binary representation. To this end, in practice, CSD ternary encoding is considered, as it allows the automatic synthesis of a filter from the provided genome while reducing circuit complexity.

In CSD representation, a radix-2 signed-digit number system is adopted. Each digit is given by $d_k \in \{-1, 0, +1\}$, and it is associated with the POT term 2^k . CSD imposes a structural constraint, namely that no two adjacent digits may be nonzero:

$$(2) \quad d_k \neq 0 \Rightarrow d_{k-1} = d_{k+1} = 0.$$

This rule guarantees a unique representation with the minimum possible Hamming weight among all radix2 signed-digit expansions. In a FIR filter, we have:

$$(3) \quad y[n] = \sum_{k=0}^N h[k] x[n - k].$$

Each multiplication becomes:

$$(4) \quad h[k]x[n - k] = \left(\sum_{i=0}^W d_{k,i} 2^{-i}\right)x[n - k] = \sum_{i=0}^W d_{k,i}(x[n - k] \gg i).$$

In other words, multiplication is simplified to an operation consisting only of simple bit shifts and additions/subtractions.

Given the aforementioned properties, in practical FIR filter design scenarios, a CSD encoding is adopted [2]. In particular, a filter coefficient is represented by a string of ternary digits. Each of the ternary digits corresponds to a POT. Let $k2^m$ be a POT, where m can be negative in the case of floating-point numbers. If k is positive (+1), the position $|m|$ of the string contains 1. If k is negative (−1), the

position $|m|$ of the string contains $\bar{1}$. If k is zero, the position $|m|$ of the string contains zero as well. For instance, consider the coefficient 0.421875, which can be written as: $2^{-1} - 2^{-4} - 2^{-6}$. In this case, the ternary encoding is 0100 $\bar{1}$ 0 $\bar{1}$ 0. The ternary string length equals the filter coefficient Wordlength W . By concatenating all the encoded coefficients, a ternary bit stream is created, corresponding to a GA chromosome.

2.2. Hardware complexity and cost savings

As previously mentioned, the cost savings from using a CSD representation come from the hardware being inherently multiplierless. Given a FIR filter of order N , wordlength W and exactly B non-zero bits per coefficient, we now showcase these savings with a simplified model for hardware complexity and cost.

Starting with the CSD encoding, for each coefficient, $B - 1$ adders are needed to accumulate the POTs. Additionally, B shifters are required to compute each POT, leading to the cost of a single coefficient amounting to:

$$C_{\text{coeff}} = BC_{\text{shift}} + (B - 1)C_{\text{add}}.$$

Accumulating the results of the $N + 1$ coefficients requires N additional adders; therefore: $C_{\text{acc}} = NC_{\text{add}}$. Thus, the total cost of the CSD implementation corresponds to:

$$(5) \quad C_{\text{CSD}} = B(N + 1)C_{\text{shift}} + [B(N + 1) - 1]C_{\text{add}}.$$

For a naïve implementation, a possible encoding could be a fixed-decimal representation where each coefficient is

$$h[k] = -b_{k,W-1}2^0 + \sum_{i=0}^{W-2} b_{k,i} 2^{i-(W-1)} b_{k,i} \in \{0, 1\}.$$

Hence, the cost would simply be

$$(6) \quad C_{\text{naive}} = (N + 1)C_{\text{mult}} + NC_{\text{add}}.$$

Regarding the costs of the single component types, in digital hardware the complexity of arithmetic operators scales differently with operand width n . A fixed bit shift can be implemented as a wiring permutation and therefore incurs negligible hardware complexity of $O(1)$. Addition requires several elementary adder cells proportional to n , resulting in $O(n)$ hardware complexity, whereas conventional binary multiplication requires the generation and accumulation of n^2 partial products, leading to $O(n^2)$ hardware complexity [28]. Of course, actual physical implementations can vary in complexity and cost, so to keep our estimates acceptably abstract, we assign the following normalized costs:

- $C_{\text{shift}} = 0$,
- $C_{\text{add}} = \frac{W}{W} = 1$,
- $C_{\text{mul}} = \frac{W^2}{W} = W$.

In the simple case of a filter of order $N = 8$, wordlength $W = 8$ and number of non-zero digits $B = 3$, we would have $C_{\text{CSD}} = 26$ whereas $C_{\text{naive}} = 80$, roughly resulting in a 67% cost reduction.

2.3. Genetic algorithms

EAs constitute a family of population-based metaheuristic optimization methods whose design is inspired by the principles of biological, Darwinian evolution. EAs do not exhaustively explore a search space, and they do not follow a deterministic gradient. Instead, EAs maintain a collection of candidate solutions (i.e., a population) and iteratively refine this collection through stochastic operators. Over successive iterations (i.e., generations), the population is biased toward regions of the search space that include better fitness (objective) function values, while retaining sufficient diversity to avoid premature convergence to suboptimal solutions.

Each candidate solution is called an individual (chromosome in GAs). The set of all individuals at a given time step is the population. EAs operate on an encoded representation of a solution. The fitness function (or objective function) encapsulates the problem to be solved and assigns a scalar quality score to each individual. Fitness evaluation is typically the most computationally expensive operation in EAs.

Selection mechanisms stochastically choose individuals for reproduction. Variation is introduced by crossover (also called recombination), which combines genetic material from two parents, and by mutation, which applies small random perturbations to a single individual. After variation, the new offspring must be integrated into the population. Generational models replace the entire population at each iteration, whereas steady-state models replace only a few individuals at a time. Elitism is the practice of unconditionally preserving the best individual(s) across generations, guaranteeing that solution quality is monotonically non-decreasing.

A GA represents solutions as fixed-length strings/vectors over a finite alphabet and applies selection, crossover, and mutation to evolve the population toward higher-fitness regions of the search space. The most straightforward GA representation is a binary string of fixed length L , where each bit corresponds to a decision variable (i.e., gene). Binary encoding is universal in the sense that any finite discrete problem can be mapped to it. Real-valued (floating-point) encoding employs continuous decision variables. Integer encoding is used for combinatorial problems such as job scheduling or routing. The choice of encoding deeply affects algorithm performance and must be guided by the problem structure.

Population initialization is typically performed by generating each gene of each individual uniformly at random within its admissible domain. For binary strings, each bit is set independently to 0 or 1 with equal probability. Fitness evaluation is problem-specific and may involve simulation, numerical integration and calculation, database lookup, or calls to external software.

Selection is the mechanism by which GAs allocate parents for crossover to create the mating pool. The most common schemes are as follows:

- Roulette wheel selection: each individual is selected with a probability directly proportional to its fitness. This operator is simple but sensitive to fitness scaling; that is, a single super-fit individual can dominate the population, causing premature convergence (a phenomenon known as genetic drift).
- Tournament selection: a subset of k individuals is drawn uniformly at random with replacement; the fittest member wins and is copied to the mating pool.

Crossover is a variation operator that combines genetic material from two (or more) parent chromosomes to produce offspring, embodying the intuition that good solutions share good partial structures:

- One-point crossover: a random cut point $c \in \{1, \dots, L - 1\}$ is chosen; offspring receive the prefix of one parent and the suffix of the other.
- Two-point and k -point crossover: generalizations that use multiple cut points, reducing positional bias, and enabling more diverse recombination patterns.
- Uniform crossover: each gene is inherited independently from one parent with a probability of 0.5, producing offspring with maximum allelic diversity.

Mutation introduces random perturbations that prevent population stagnation and enable the GA to escape local optima. The mutation rate (typically $1/L$ for binary GAs) must be set carefully: if too low, diversity is hindered; if too high, the search degenerates toward a random walk:

- Bit-flip mutation (binary): each bit is flipped independently with probability equal to the mutation rate.
- Gaussian mutation (real-valued): a zero-mean Gaussian deviate with standard deviation σ is added to each gene.
- Uniform mutation (real-valued): a gene is replaced by a uniform random value within its domain.

In Fig. 1, we show an example of a vanilla GA workflow.

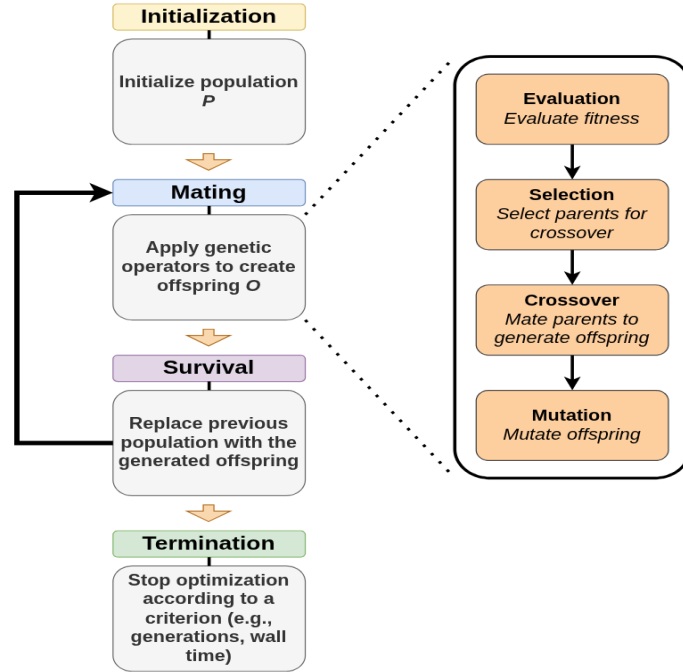


Fig. 1. Example of vanilla GA workflow

3. Methodology

The most straightforward way of representing an individual composed of the coefficients of an FIR filter in GA is by using real-valued fixed-length vectors, whose length is given by the number of coefficients $N + 1$ of the filter (order N), even if it is not actually a practical approach. However, as we said in Section 2, in practical scenarios, real-valued coefficients cannot be employed to implement digital filters in hardware. In fact, representations based on real numbers cannot be used since they require potentially infinite precision that cannot be implemented in hardware. Any approximation using a floating-point number, as done with GA, will have to deal with limited precision and, for hardware, a more complex implementation than for fixed-point numbers. Furthermore, such a representation does not allow us to verify or control CSD constraints, which are of paramount importance for implementing filters that incur low cost [2]. Nonetheless, we still consider a GA method with real-valued coefficients an ideal baseline for a proof-of-concept and a solid basis for comparison with other approaches. We call this baseline “Real-valued GA”.

The real-valued baseline initializes each individual by uniformly sampling each value from $[-1, 1]$. It employs roulette wheel selection, intermediate crossover, and Gaussian (non-adaptive) mutation. As we will see in the experimental phase,

we test this baseline using both tournament selection and roulette wheel selection. Along with the real-valued baseline, we assess the effectiveness of the “Basic CSD GA” method from Lee et al. [2]. By taking inspiration from this work, we propose an alternative GA strategy, named “Enhanced CSD GA”, that aims to mitigate the limitations of the original algorithm.

The original paper employs the CSD-based ternary encoding presented in Section 2. Each individual is initialized with one non-zero bit per coefficient ($h = 2 - k$), and roulette-wheel selection is used. Regarding crossover and mutation, the original paper employs two-point crossover and digit-flip mutation, which extends the standard bit-flip mutation to a ternary set.

The main design constraints are represented by the wordlength W , which is the number of bits our hardware can handle, and the maximum number of non-zero digits in a coefficient (N_{digits}). Having many non-zero digits implies a higher circuit complexity. Since the crossover and mutation operators employed in [2] often violate CSD constraints and exceed the allowed maximum N_{digits} , we modify the “Basic CSD GA” by implementing novel crossover and mutation operators. We call these new genetic operators CSD-preserving crossover and mutation.

The CSD-preserving crossover is a two-point crossover in which the crossover points are chosen from values that are multiples of W . This ensures that the crossover occurs on coefficient boundaries, thereby swapping CSD-compliant coefficients in their entirety. The CSD-preserving mutation consists of the standard digit-flip mutation with a repair step. In particular, the repair step is performed whenever adjacent non-zero digits are generated or in case the N_{digits} constraint is broken, as detailed in the following pseudo-code:

Algorithm 1 CSD-preserving mutation step

```

1: if  $d_i \neq 0$  then
2:   flip digit  $d_i$ 
3: else
4:   for all  $j \in \{i - 1, i + 1\}$  s.t.  $0 \leq j < W$  and  $d_j \neq 0$  do
5:      $d_j \leftarrow 0$ 
6:   end for
7:   if  $\|\mathbf{d}\|_{\text{nonzero}} = N_{\text{digits}}$  then
8:      $k \leftarrow \max\{j \mid d_j \neq 0\}$   $\triangleright$  least significant nonzero digit
9:      $d_k \leftarrow 0$ 
10:  end if
11:  flip digit  $d_i$ 
12: end if

```

Fig. 2. Pseudo-code for the CSD-preserving mutation step

Our new version, just like the original version and the real-valued baseline, employs roulette wheel selection. However, we also test all the methods described with tournament selection. The elitism of the best-so-far is always enforced across all methods. The rest of the algorithm remains unchanged, allowing us to isolate the effect of genetic operators.

We use the minimax error employed in [2] as the fitness function, which searches for the maximum peak error throughout a discrete frequency domain via the magnitude response difference between the target and designed filters:

$$(7) \quad E_{L_\infty} = \max ||H_T(\omega, \gamma) - H_D(\omega, \gamma)||,$$

where $H_T(\omega, \gamma)$ and $H_D(\omega, \gamma)$ are respectively defined as the frequency responses of the Target and Designed filters. The frequency response is obtained from the transfer function (Equation (1)) by substituting $z = e^{j\omega T}$ as follows:

$$(8) \quad H(\omega, \gamma) = \sum_{n=0}^N a_n e^{-jn\omega T},$$

where T and ω are the sampling period and angular frequency, respectively.

4. Experiments and results

In our experimental analysis, we seek to answer the following research questions:

- **RQ1:** Can our method discover CSD-compliant digital filters with higher quality than those obtained by the baseline?
- **RQ2:** Is it true that the ideal real-valued baseline consistently provides better filters?
- **RQ3:** Does the selection operator have an impact on the quality of the solutions?

In all our experiments, we set the population size to 100 and evolved it for 300 generations with 5% elitism. Crossover is always performed, while mutation, for each gene, has a probability inverse to the genome's length. The offspring always replaces the entire population at the end of each generation. This way, the offspring becomes the population for the next generation.

We explore orders $N \in \{8, 16, 24, 32\}$ and wordlengths $W \in \{10, 12, 14, 16\}$. We set $N_{\text{digits}} = 3$ (maximum number of non-zero digits in a coefficient) and $N_{\text{max}} = 100$ (maximum number of retries when generating a CSD-invalid individual during crossover and mutation, only applies to the "Basic CSD GA"). The fitness is computed by uniformly sampling the frequency domain with 4096 samples. Since [2] does not report all hyperparameter settings, we test several values of N and W ; $N_{\text{digits}} = 3$ is set to a low value according to domain knowledge from previous literature $N_{\text{max}} = 100$. This provides a sufficient number

of retries for the Basic CSD GA. This should allow us to execute a strong baseline and have a fairer comparison with our method.

Our experimental campaign consists of testing the three outlined methods on five target filters: lowpass easy, lowpass medium, lowpass sharp, lowpass wide, and lowpass narrow. The first three targets are variations on an ideal lowpass filter centered at $\omega = 1.5$ in which the transition band, which is left to the GA to interpolate, is respectively: $[1.1, 1.9]$, $[1.3, 1.7]$, and $[1.4, 1.6]$. The last two filters are ideal lowpass filters centered respectively at $\omega = 0.85$ and $\omega = 2$ with transition bands $[0.7, 1.0]$, and $[1.8, 2.2]$. For each method, filter, order, and wordlength, we perform 30 independent repetitions.

To ensure the statistical validity of the observed behaviors, we compared results using a pairwise Mann-Whitney U test [29] and, when comparing multiple methods, applied a Holm-Bonferroni correction [30]. Before pairwise testing, we performed a Kruskal-Wallis test [31] to ensure at least one method is significantly different. All tests are conducted with $\alpha = 0.05$. Statistical tests are carried out by accounting for the error of the final solutions. The Python implementation of all the code used in this work is available online at <https://github.com/simdeistud/optimization-ai/tree/paper-benchmarks>.

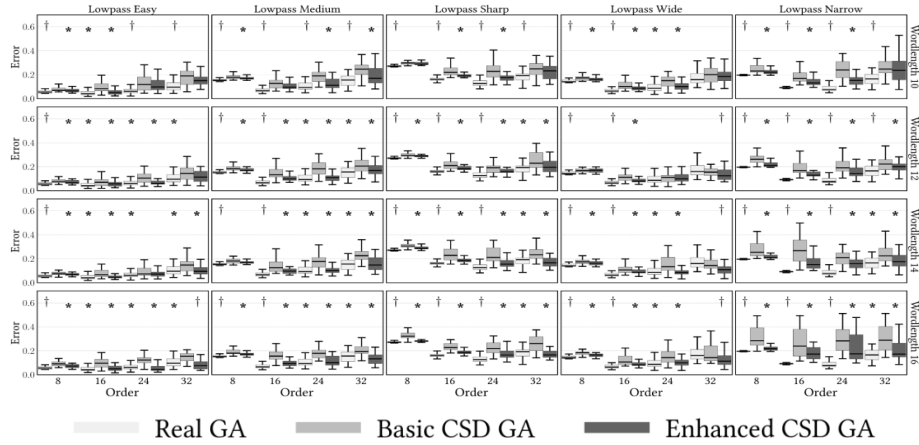


Fig. 3. Boxplot showing the error distribution of the best solutions discovered by the three methods across 30 repetitions with roulette wheel selection. For each group given by order, filter, and wordlength, we highlight with an asterisk (*) the methods that outperform at least one other method in the group. Moreover, we highlight the method that outperforms all the others in the group with a dagger

The experimental results related to the methods executed with roulette wheel selection (the one selection operator employed in [2]) are presented in Fig. 3. In all 80 configurations ($5 \text{ filters} \times 4 \text{ orders} \times 4 \text{ word lengths}$), our method achieves equal or lower median error than the Basic CSD GA. Additionally, our method

consistently outperforms the method of [2] while also preserving CSD constraints, making it a more robust and reliable approach. The real-valued baseline achieves the lowest errors in most configurations, but its solutions are not hardware-implementable. For smaller values of the order, the ternary encoding-based methods generally become more stable.

A notable case is the lowpass wide filter with order 32 and wordlength greater than or equal to 12, when our method clearly outperforms both the reference baseline and the original approach. That the Enhanced CSD GA can outperform the real-valued baseline may seem counterintuitive, but two factors explain this result. This result can be attributed to two factors. First, at order 32, the real-valued GA operates in a 33-dimensional continuous space, where the search becomes harder within the allotted 300 generations. Second, the CSD encoding implicitly constrains the search space to a discrete, structured subset of coefficient values. This acts as a form of regularization that can help the evolutionary search avoid poor local optima in higher-dimensional problems. In other words, the CSD representation trades off precision for a more navigable search landscape, and this trade-off becomes favorable as the filter order increases.

In addition, we repeat all the previous experiments by using tournament selection with tournament size 4.

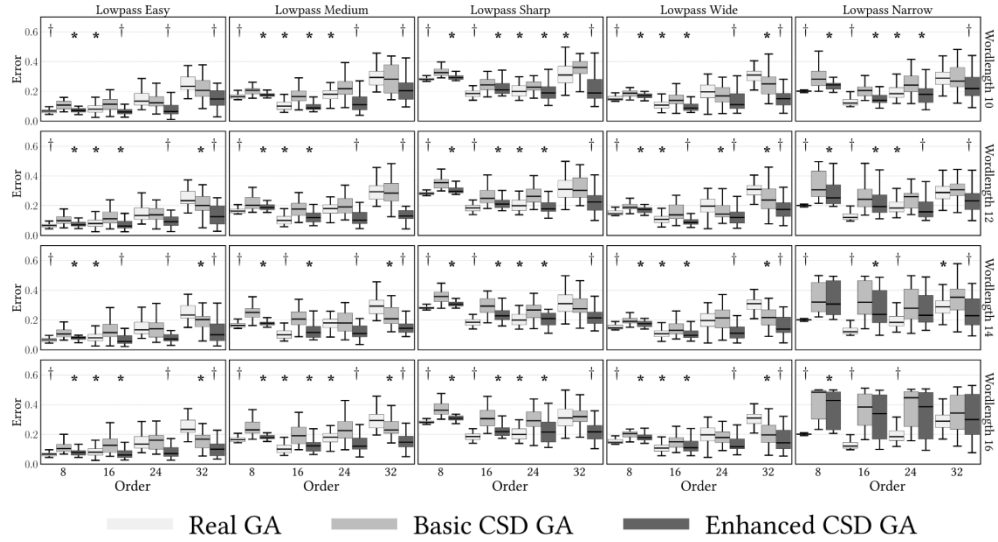


Fig. 4. Fig. 4. Boxplot showing the error distribution of the best solutions discovered by the three methods across 30 repetitions with tournament selection. For each group given by order, filter, and wordlength, we highlight with an asterisk (*) the methods that outperform at least one other method in the group. Moreover, we highlight the method that outperforms all the others in the group with a dagger

The experimental results related to the methods executed with tournament selection are presented in Fig. 4. As we can see, consistently with the analysis depicted in Fig. 3, our method is never worse than the original approach in [2], and in several cases we are able to outperform the original baseline while ensuring CSD constraints are satisfied. As expected, the real-valued baseline still obtains lower errors in most scenarios. Notably, with order 32, the real-valued baseline is often worse than the other approaches. This is consistent with the results observed under roulette wheel selection (Fig. 3), where the real-valued baseline also degrades at order 32. Across both selection strategies, the CSD-based methods show greater robustness at higher filter orders, suggesting that the discrete structure of the CSD encoding provides a structural advantage when the dimensionality of the search space increases.

As a final analysis, we compare the two selection operators, roulette and tournament, within our proposed technique.

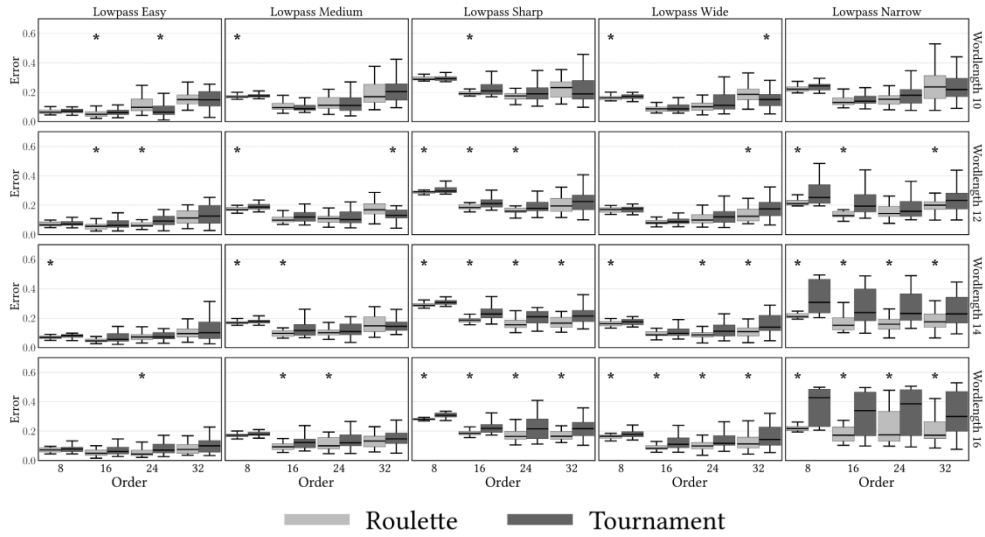


Fig. 5. Boxplot showing the error distribution of the best solutions discovered by our “Enhanced CSD GA” method with the two tested selection strategies across 30 repetitions. For each group given by order, filter, and wordlength, we highlight with an asterisk (*) the best selection operator

The experimental results comparing the two selection strategies are presented in Fig. 5. From the plot, it seems the impact of the selection strategy does not have a clear trend. The roulette wheel selection performs surprisingly well. In particular, it performs at least as well as the tournament in most cases, with exceptions including: easy filters with order 24 and wordlength 10, medium filters with order 32 and wordlength 12, and wide filters with order 32 and wordlength 10. The cases where tournament selection outperforms roulette wheel selection do not follow a

clear pattern across filter types or wordlengths. This suggests that the advantage of tournament selection, when present, is configuration-specific rather than systematic. Overall, the choice of selection operator appears to have a smaller effect on performance than the choice of genetic operators (crossover and mutation), which is the main focus of this work.

5. Discussion

After our analysis, we try to provide a concrete answer to our previously defined three research questions.

- **RQ1:** Can our method discover CSD-compliant digital filters with higher quality than those obtained by the baseline?

From our analysis, we can safely state that our method outperforms the original approach presented in [2], while guaranteeing that CSD constraints are satisfied, enabling a cost-effective and easier practical hardware implementation. This is consistent across all tested filters, orders, and wordlengths, where no single case in which our method is statistically worse than the baseline is recorded.

- **RQ2:** Is it true that the ideal real-valued baseline consistently provides better filters?

As expected, the real-valued baseline provides better filters in most cases. However, as already discussed in the manuscript, these filters cannot be practically implemented in hardware. Surprisingly, we observe some notable cases in which our method is better than the ideal baseline. Even though this may seem deceptive, running a GA with real-valued coefficients does not guarantee that a better solution is always found.

- **RQ3:** Does the selection operator have an impact on the quality of the solutions?

After analyzing the two selection operators employed, we found that previous findings are confirmed regardless of the selection strategy, meaning that our method still outperforms the baseline even when the selection operator is switched from roulette wheel selection to tournament selection. Regarding the comparison with the two selection operators themselves, it seems that roulette wheel selection is more effective. However, apparently, there is not a clear pattern, suggesting that the selection operator itself does not meaningfully impact the evolution in this specific domain scenario.

Our analysis highlights that our approach is effective and practical for hardware implementations.

6. Conclusion

We presented an Enhanced CSD GA for the automatic synthesis of FIR digital filters with hardware-efficient CSD coefficients. Our main contribution consists of two modified genetic operators, a coefficient-level crossover and a mutation with a repair step, that guarantee CSD compliance by construction, eliminating the retries and coefficient resets required by the original approach of Lee et al. [2].

Our experimental results across five lowpass filter targets, four filter orders, and four wordlengths show that the Enhanced CSD GA consistently matches or outperforms the Basic CSD GA, regardless of the selection operator used. Notably, at higher filter orders ($N = 32$), the CSD-based methods prove more robust than the real-valued baseline, suggesting that the discrete structure of CSD encoding provides an advantage in higher-dimensional search spaces. This makes our approach especially suitable for practical scenarios where both filter complexity and hardware cost must be controlled.

Several directions remain open for future work. First, an ablation study isolating the individual effect of the CSD-preserving crossover and mutation would clarify which operator contributes more to the observed improvements. Second, adaptive strategies for tuning mutation probability and tournament size during the evolutionary run could further improve convergence. Finally, alternative encodings that reduce chromosome length, for instance, by exploiting filter symmetry in linear-phase designs, could make the approach scalable to higher-order filters.

References

1. Chen, X., T. Parks. Design of FIR Filters in the Complex Domain. – IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. **35**, 1987, No 2, pp. 144-153. DOI: 10.1109/TASSP.1987.1165111.
2. Lee, A., M. Ahmadi, G. A. Jullien, W. C. Miller, R. S. Lashkari. Digital Filter Design Using Genetic Algorithm. – In: Proc. of IEEE Symposium on Advances in Digital Filtering and Signal Processing. Symposium (Cat. No 98EX185), Victoria, BC, Canada, 1998, pp. 34-38. DOI: 10.1109/ADFSP.1998.685690.
3. Hsia, T. C. A Technique for Synthesizing Digital Filters. – IEEE Transactions on Instrumentation and Measurement, Vol. **18**, 1969, No 2, pp. 93-96. DOI: 10.1109/TIM.1969.4313774.
4. Saramaki, T., T. Neuvo, S. K. Mitra. Design of Computationally Efficient Interpolated FIR Filters. – IEEE Transactions on Circuits and Systems, Vol. **35**, 1988, No 1, pp. 70-88. DOI: 10.1109/31.1701.
5. Karam, L. J., J. H. McClellan. Complex Chebyshev Approximation for FIR Filter Design. – IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. **42**, 1995, No 3, pp. 207-216. DOI: 10.1109/82.372870.
6. Mitchell, M. An Introduction to Genetic Algorithms. MIT Press, 1998. DOI: 10.7551/mitpress/3927.001.0001.
7. Gen, G., M. L. Lin. Genetic Algorithms and Their Applications. London, Springer London, 2023, pp. 635-674. DOI: 10.1007/978-1-4471-7503-2_33.

8. Goldberg, D. Genetic Algorithms in Search, Optimization, and Machine Learning. United States, Addison-Wesley Longman Publishing Co., 1989. DOI: 10.5555/534133.
9. Holland, J. Genetic Algorithms and the Optimal Allocation of Trials. – SIAM Journal on Computing, Vol. **2**, 1973, No 2, pp. 88-105. DOI: <https://doi.org/10.1137/0202009>.
10. Holland, J. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, 1992. DOI: 10.5555/531075.
11. Cemes, R., D. Ait-Boudaoud. Genetic Approach to Design of Multiplierless FIR Filters. – Electronics Letters, Vol. **29**, 1993, Issue 24, pp. 2090-2091. DOI: 10.1049/el:19931396.
12. Suckley, D. Genetic Algorithm in the Design of FIR Filters. – IEEE Proceedings – Circuits, Devices and Systems, Vol. **138**, 1991, No 2, pp. 234-238. DOI: 10.1049/ip-g-2.1991.0043.
13. Wade, G., A. Roberts, G. Williams. Multiplier-Less FIR Filter Design Using a Genetic Algorithm. – IEEE Proceedings – Vision, Image and Signal Processing, Vol. **141**, 1994, Issue 3, pp. 175-180. DOI: 10.1049/ip-vis:19941185.
14. Vandeva, E. Multi-Objective Genetic Modified Algorithm (MOGMA). – Cybernetics and Information Technologies, Vol. **12**, 2012, No 2, pp. 23-33.
15. Fuller, A. T. G., B. Nowrouzian. A Novel Technique for Optimization over the Canonical Signed-Digit Number Space Using Genetic Algorithms. – In: Proc. of IEEE International Symposium on Circuits and Systems (ISCAS'2001), Vol. **2** (Cat. No 01CH37196), Sydney, NSW, Australia, 2001, pp. 745-748. DOI: 10.1109/ISCAS.2001.921178.
16. Mercier, P., B. Nowrouzian. Design of FRM Digital Filters over the CSD Multiplier Coefficient Space Employing Genetic Algorithms. – In: Proc. of IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, France, 2006. DOI: 10.1109/ICASSP.2006.1660817.
17. Kilambi, S. M., B. Nowrouzian, Y. Wu. A Novel Diversity-Controlled Genetic Algorithm for Rapid Optimization of Bandpass FRM FIR Digital Filters Over CSD Multiplier Coefficient Space. – Circuits Syst. Signal Process, Vol. **27**, 2008, pp. 599-625. DOI: 10.1007/s00034-008-9045-8.
18. Gupta, G. S., D. S. Acharya, P. Mishra, V. H. Shah. A Comparative Assessment of Evolutionary Algorithms for Design of Optimal FIR Filters. – In: Proc. of International Conference on Communication and Smart Devices (ICCoSD'25), Ranchi, India, 2025, pp. 1-6. DOI: 10.1109/ICCoSD66074.2025.11348549.
19. Gupta, S., B. Agarkar. A Review of Multiobjective Digital Filter Design Using Optimization Algorithms. The Future of Business and Society: Integrating Sustainable Technologies with Data-Driven Solutions. First Ed. CRC Press, 2026, DOI: 10.1201/9781003778882.
20. Rajasekhar, K. L₁-Norm and LMS-Based Digital FIR Filters Design Using Evolutionary Algorithms. – J. Electr. Eng. Technol., Vol. **19**, 2024, pp. 753-762. DOI: 10.1007/s42835-023-01589-7.
21. Lian, L., Z. Tian. FIR Digital Filter Design Based on Improved Artificial Bee Colony Algorithm. – Soft Comput., Vol. **26**, 2022, pp. 13489-13507. DOI: 10.1007/s00500-022-07506-w.
22. Samiappan, R. A. S., M. Prabukumar. Fine-Tuning Digital FIR Filters with Gray Wolf Optimization for Peak Performance. – Sci. Rep., Vol. **14**, 2024, 12675. DOI: 10.1038/s41598-024-62403-6.
23. Srivatsan, K., N. Venkatesan. Improved Meta-Heuristic Technique to FIR Filter Design and Application. – In: IEEE Access, Vol. **12**, 2024, pp. 108097-108107. DOI: 10.1109/ACCESS.2024.3431444.
24. Bansal, P., S. G. Sandeep. Optimal High Pass FIR Filter Based on Adaptive Systematic Cuckoo Search Algorithm. – Cybernetics and Information Technologies, Vol. **22**, 2022, No 4, pp. 167-177.
25. Boudjelaba, K., et al. An Efficient Hybrid Genetic Algorithm to Design Finite Impulse Response Filters. – Expert Syst. Appl., Vol. **41**, 2014, pp. 5917-5937. DOI: 10.1016/j.eswa.2014.03.034.

26. Singh, S. Optimum FIR Filter Design Using Dynamic Differential Evolution with Hybrid Mutation Strategy. – In: S. J. Nanda, R. P. Yadav, M. Prasad, M. Saraswat, Eds. Data Science and Applications. ICDSA 2025. – In: Lecture Notes in Networks and Systems. Vol. **1795**. Cham, Springer, 2026. DOI: 10.1007/978-3-032-15404-0_34.
27. Yu, Z., F. Zhu, Y. Yu, Y. Chen, J. Ye, X. Yang, J. Yang. Design of Low Complexity Digital FIR Filter Based on PSO-LSTM. – Engineering Letters, Vol. **34**, 2026, No 5, pp. 1700-1706.
28. Parhami, B. Computer Arithmetic: Algorithms and Hardware Designs. Second Edition. Chapters 2-3. New York, Oxford University Press, 2010.
29. Mann, H. B., D. R. Whitney. On a Test of Whether One of Two Random Variables is Stochastically Larger Than the Other. – The Annals of Mathematical Statistics, Vol. **18**, 1947, No 1, pp. 50-60.
<https://www.jstor.org/stable/2236101>
30. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. – Scandinavian Journal of Statistics, Vol. **6**, 1979, No 2, pp. 65-70.
<https://www.jstor.org/stable/4615733>
31. Kruskal, W., W. Allen Wallis. Use of Ranks in One-Criterion Variance Analysis. – In: Journal of the American Statistical Association. Vol. **47**, No 260, 1952, pp. 583-621. DOI: 10.1080/01621459.1952.10483441.

Fast-track. Received: 01.07.2026; First Revision: 29.07.2026, Second Revision: 10.08.2026;

Accepted: 15.08.2026

© 2026 by the authors. Licensee IICT-BAS, Sofia, Bulgaria. This article is an open access article distributed under the terms and conditions of the **Creative Commons Attribution-NonCommercial-NoDerivatives** license.