Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 1022 – 1033

doi:10.3906/elk-1311-259

Speciation-based genetic algorithm in analog circuit design

Hasari KARCİ[1,*], Gülay TOHUMOĞLU[2], Arif NACAROĞLU[1]

[1]Department of Electrical and Electronics Engineering, Faculty of Engineering, Gaziantep University, Şehitkamil, Gaziantep, Turkey

[2]Department of Electrical and Electronics Engineering, Faculty of Engineering, Dokuz Eylül University, Tınaztepe Campus, Buca, İzmir, Turkey

Received: 29.11.2013 • **Accepted/Published Online:** 31.01.2014 • **Final Version:** 23.03.2016

Abstract: This paper presents a speciation procedure that improves the local search capability of the genetic algorithm in analog circuit design. There is no need for additional circuit simulation in order to apply this procedure. The procedure is tested in Gaussian, sigmoid, cube, and square circuit design problems. Two sets of 125 simulations with the same seed values are performed for each problem using both the proposed procedure and the canonical genetic algorithm. The simulation results show that the method is statistically better than the canonical genetic algorithm, which suffers from bad locality. The effects of the population size and speciation threshold coefficient on the performance of the speciation algorithm are investigated. Confidence intervals of the simulation results are calculated. The results show that the speciation procedure improves the quality of solutions with at least 99% confidence, and the effectiveness of the method, which is statistically determined, increases in small populations.

Key words: Genetic algorithm, genetic programming, analog circuit design, speciation, local search

1. Introduction

Automatic design of electronic circuits is a challenging engineering problem. Although automatic design of digital circuits has made great progress, most analog circuits are still designed by skilled engineers, and their methods are mainly based on domain-specific knowledge. Fortunately, today's computer technology and circuit theory make the use of unconventional methods in analog circuit design possible. The genetic algorithm (GA) is used as a search algorithm in different analog circuit design problems successfully [1–5]. Though the GA is inspired by natural evolution, it is hard to mimic all dynamics of natural evolution in the GA. One of these dynamics is speciation.

Speciation in nature can be regarded as a local search within natural evolution. In nature, individuals of the same species mate to produce new generations. The offspring's genes come from its parents; during the crossover operation in meiosis, its genes are aligned and the genes that are responsible for the same characteristics are exchanged. As a result, the offspring's genes are slightly different from its parents'. Generally, these slight differences in genotype produce minor differences in the phenotype of the offspring. Thus, offspring characteristics are close to each other and to those of their parents. This mechanism yields a good locality and can be considered a local search in the species region. Therefore, it is possible to effectively search local regions of the species. Good locality is important for an effective search algorithm [6].

*Correspondence: karci@gantep.edu.tr

The canonical GA (GA-c) suffers from bad locality in analog circuit design [7]. In the GA-c, the topology of the circuit and component values are constructed according to the genotypes of individuals. Unlike in nature, a small change in the topology of a circuit may drastically change the electrical behavior of circuit, or in other words, a circuit's phenotype. Therefore, two topologically close circuits may not have similar electrical behavior. Since the fitness of the individuals is computed according to their electrical behaviors, their fitness values can be very different. Besides this, two circuits that have the same fitness value may have very different circuit topology. For the sake of clarification, consider the circuits shown in Figure 1a–1c. Although it is a very simple example, it clearly exhibits a bad locality problem.

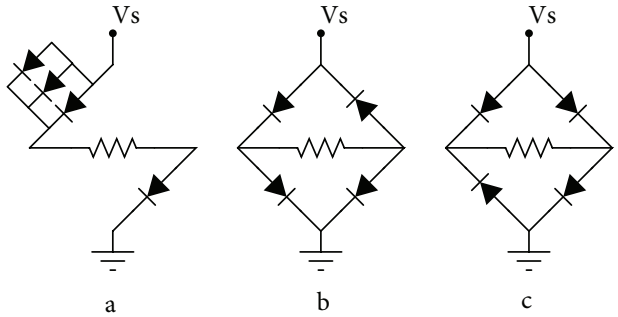

Figure 1. Half-wave rectifier circuits.

Three half-wave rectifier circuits are shown in Figure 1. Though Figure 1c is topologically very different from Figure 1a and b, their phenotype and fitness values are the same. There are many such examples in analog circuit design [8,9]. If one wants to design a full-wave rectifier circuit with the GA, these types of circuits can emerge in the population. Obviously, a mate between Figure 1a and 1b can produce an exact solution, but a crossover between Figure 1c and 1b or 1a may result in a macromutation. The good building blocks in Figure 1a and 1b can be destroyed with this mating.

In a GA-c implementation, while the population evolves, it is expected that all individuals converge to the best individual, topologically, due to selection pressure. Most of the individuals become alike in phenotype and genotype. These slightly different individuals search the local region around the best individual. However, the aforementioned problems make it difficult to perform a local search when the GA-c is used in analog circuit design.

A speciation procedure can alleviate the bad locality problem of the GA-c. In the literature, speciation methods and hybrid GA methods, which combine the GA with local search methods, are used to increase the local search capability of the GA. Speciation methods are already used in digital circuit design. For example, Hwang [10] used a speciation method based on fitness sharing to solve a digital circuit and showed that the GA with speciation (GA-s) has a higher convergence rate. Hwang's method finds exact solutions at a higher rate. There is no application of the GA-s in analog circuit design in the literature. In [11], an immunity-based GA was used in VLSI floorplan design problems. It divides a population into several subpopulations and controls similar solutions in order to balance between local search and global search and to prevent premature convergence. It improves the average performance of the best individual by up to 10% over GA-c, but also increases simulation time by about 47%. Another VLSI floorplan optimization method that combines greedy search with the GA was proposed in [12]. During the local search, gene segments of the individuals are swapped, and fitness of individuals are recalculated. It was tested on 5 different problems. It improved the best solutions by up to 0.83% according to GA-c. Song [13] proposed a new GA called the improved GA that uses five different optimization methods,

including module crossover, fitness sharing, double selection population, queen bee mating, and exponential weighting to improve the performance of the GA-c. The aforementioned methods and improved GA method were applied to a 1-bit full adder design problem. The results were compared with the GA-c, and it revealed that the improved GA drastically reduces the evolutionary time and average convergence and increases the success rate. Although the method was tested on only one problem, its performance over the GA-c is impressive. In [14], a two-stage optimization method, which consisted of the GA and a pattern search algorithm, was proposed to optimize induction motor parameters. In the first cycle, the GA-c finds an optimal solution, and then this solution is used as the initial point of a pattern search algorithm. The experimental results showed that the pattern search algorithm slightly improved the solution found by the GA-c. Lo [15] combined the shooting local search method and differential evolution with the GA-c to optimize panel gate driver circuits with an amorphous silicon transistor. During evolution, the population was divided into two subpopulations according to individual fitness values. While the shooting local search was performed on the first half of the population, differential evolution was applied to the other half of the population. Thus, this method increases the number of circuit simulations. The method was applied to two different driver circuits, and the best solutions were compared with the best solutions found with the GA-c. It improved the final solution by 21% and 10%. A local search hybridized GA was used to design combinational logic circuits in [16], and it was compared with the GA-c. The results showed that the hybridized GA improves the average fitness of final solutions. However, it almost doubles the number of circuit evaluations due to the local search. Evolutionary programming was used to identify effective search regions in [17]. The effective regions are investigated with a gradient search method. If the local search does not find an acceptable solution, that region is labeled as a forbidden region. Inevitably, this method requires additional circuit simulations to perform local searches. Actually, it is unfair to compare the GA-c with a method that requires extra circuit simulations. The population size of the GA-c must be increased proportionally for a fair comparison. Huang et al. used an orthogonal local search GA to optimize the component values of power electronic circuits [18] and the method was compared with the GA-c. Although the proposed method does not need additional circuit simulations, it slightly improves the final solution. In analog circuit design, circuit simulation consumes the most time and computation power, so it is essential to find a method that does not need additional circuit simulations.

In this paper, a speciation procedure that needs no additional circuit simulations is proposed to enhance the local search capability of the GA-c. The effectiveness of the procedure is tested in four different circuit design problems, and the simulation results show that the proposed method is statistically better than the GA-c.

The organization of the paper is as follows: in Section 2, the GA-s is presented in detail. The circuit representation and simulation parameters of the GA are given in Section 3. The experimental results are discussed in Section 4, followed by conclusions.

2. Speciation procedure in the GA

In this section, a new speciation method is introduced to improve the local search capability of the GA. In the proposed method, the best fitness value of the population is recorded at the third generation and then multiplied by the threshold coefficient to find the threshold value of speciation. Then, during evolution, the best fitness value of the population is observed and compared with the threshold value. Whenever the fitness of the best individual reduces to below the threshold value, the speciation process takes place, and then the threshold value is updated. Intuitively, it is assumed that building blocks that improve the performance of the circuits have already emerged when the best fitness value is below the threshold value. Therefore, a local search around these building blocks may give better results. The GA-s algorithm is illustrated in Figure 2.

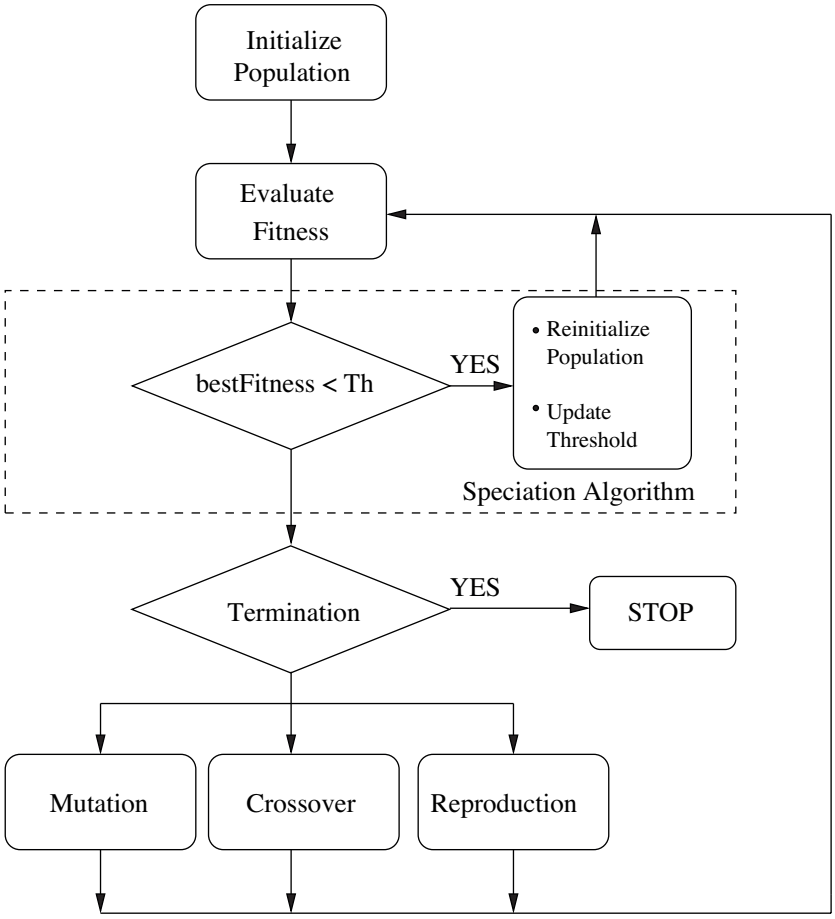

Figure 2. GA with a speciation algorithm.

In the speciation procedure, individuals are sorted according to their fitness values. The first 10 individuals are chosen as template individuals who are supposed to contain good building blocks, and then the rest of the population is deleted. The new population is reinitialized according to the chosen individuals. The number of template individuals is chosen according to the simulation results. A relatively small number of template individuals decreases the genetic diversity of the reinitialized population. On the other hand, a large number of template individuals may waste computation resources, since, hypothetically, some of these extra individuals may not contain good building blocks.

In the first step of the reinitialization process, an exact copy of the template individual is produced. Then terminal connections and component values of one component are changed randomly. Hence, the new individuals are topologically close to their template individuals. They contain the same building blocks of the template individuals and also have slightly different electrical connections. Therefore, the search space region that is close to those template individuals can be searched. If the reinitialized population can find a better solution in that search space, the speciation process takes place again, and so on.

It is obvious that the quality of the building blocks exploited by reinitialized individuals is related to the threshold value. Lower threshold values give better building blocks that improve the final solution. However, there is a contradiction between the threshold value and the occurrence number of the speciation procedure. Since the lower threshold values impose higher pressure on the best fitness value, the occurrence probability of the speciation algorithm decreases. In fact, speciation never takes place in some cases. The simulation

results show that a threshold coefficient τ between 0.6 and 0.9 is plausible. Unlike other local search methods, observation of the best fitness value to discover building blocks and the subsequent reinitialization process do not introduce a remarkable computation burden.

3. Circuit representation and simulation parameters

The design of analog circuits requires specification of the circuit topology, including the specification of the component type, parameters, and its connectivity.

In this study, fixed-length list representation was constructed to describe circuit topology. Each node of the list contains a structure named eComp that includes all information to connect a MOSFET to the circuit. The structure of eComp is shown below.

```
eComp{
   int DCto, GCto, SCto;
   int Dt, Gt, St;
   int Type;
   float W,L;
}
```

The variables DCto and Dt together specify the connection of the drain terminal of the MOSFET. While DCto points to which node will be used for connection, Dt specifies the terminal to be connected. Likewise, (GCto, Gt) and (SCto, St) stand for the gate and source terminal of the MOSFET, respectively. DCto, GCto, and SCto values are chosen randomly in intervals of $[0,\ circuitSize]$. Since Dt, Gt, and St parameters specify terminal connections of the MOSFET, their values are chosen in the set T defined as:

$$\mathrm{T} = \{\mathrm{drain},\ \ \mathrm{gate},\ \ \mathrm{source},\ \ \mathrm{ground},\ \ \mathrm{input},\ \ \mathrm{output},\ \ \mathrm{vdd}\}. \tag{1}$$

The type of the MOSFET is chosen from the fundamental MOSFET type:

$$\mathrm{Type} = \{\mathrm{nmos},\ \ \mathrm{pmos},\ \ \mathrm{nrmos},\ \ \mathrm{prmos}\}. \tag{2}$$

W and L are the channel values of the MOSFET, and they are chosen in $[2,\ 20]\ \mu$m.

The elementary circuit components are chosen as building blocks of the circuit (Figures 3a–3d). The n-channel and p-channel MOSFETs, as illustrated in Figure 3a and 3c, and the resistor implementation of the MOSFET shown in Figure 3b and 3d are used as circuit elements.

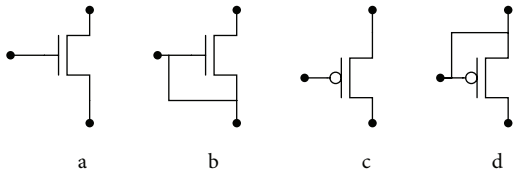

Figure 3. Building blocks of a circuit.

The components given in Figure 3 are fundamental and have no relation to the design problems. They are intentionally chosen to clearly demonstrate the contribution of the speciation algorithm to GA.

The circuit template used in the design problems is shown in Figure 4. Note that r_s and R_L resistors stand for source and load resistors, respectively. The voltage on output resistor V_{out} is used for fitness calculation.

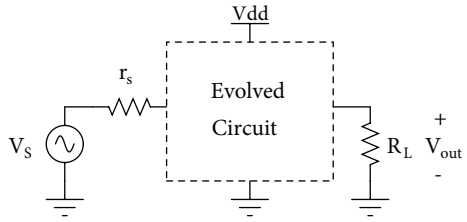

Figure 4. Embryonic circuit.

All simulations were run on computer clusters of TÜBİTAK ULAKBİM. galib247 (http://web.mit.edu/) was used to write the GA-s and GA-c algorithms. galib247 is a C++ library of the GA that includes tools for applying the GA using a number of encoding schemes and genetic operators. One-point crossover was adopted to get homolog crossover between parents. In the crossover operation, a random point is chosen in the genome. Then it is exchanged with a corresponding gene from the other parent. The swap mutation and tournament selection operators, which are built-in operators of the galib247, are chosen. The number of generations is chosen for the termination criterion. The GA parameters used in the experiments are listed in the Table.

Table. The simulation parameters of GA for the experiments.

Population size	2000
Number of generations	100
Probability of crossover	0.95
Probability of mutation	0.5

Since the GA-s and GA-c can solve the cube circuit problem for a population size of 2000, the population size of the cube circuit was reduced to 300 for the sake of comparison.

4. Experiments and results

In order to verify the effectiveness of our approach, four different analog circuits (sigmoid, Gaussian, cube, and square circuits) were synthesized. The output voltage of the evolved circuits was compared with the analytic expression of the desired output voltage, and the absolute error was assigned as a fitness value. The analytic expressions of the desired outputs are as follows:

- Gaussian circuit: $V_{out} = 0.1e^{-2.5V_{in^2}}$,
- Square circuit: $V_{out} = 0.08\ V_{in^2} + 0.01$,
- Sigmoid circuit: $V_{out} = \frac{0.08}{1+e^{-5V_{in}}}$,
- Cube circuit: $V_{out} = 0.08\ V_{in^3}$.

The fitness function is given by the following equation:

$$fitness = \sum_{i=1}^{41} |D(i) - V_{out}(i)|, \tag{3}$$

where i is the index of the sampling points of the voltage source. The input voltage interval $[-1, 1]$ is sampled for 41 points, and the desired output voltage $D(i)$ is calculated. The total absolute error between $V_{out}(i)$ and $D(i)$ is assigned as the fitness value of the individual.

Since the GA has no detailed mathematical background yet, a statistical approach was used to show the superiority of the GA-s over the GA-c. The random number generator in galib247 was seeded with the set of numbers for each simulation. Using the same seed values guarantees that the initial populations of the GA-c and GA-s are the same and that their populations will be identical until the first speciation procedure takes place. If the speciation procedure does not take place, the GA-s ends exactly the same as the GA-c simulations that use the same seed value. One hundred and twenty-five simulations were performed for each design problem using both the GA-c and GA-s. In each trial, the fitness of the best circuit was recorded. The average of the best fitness values, their standard deviation, and the confidence intervals (CIs) are calculated as:

$$x_m = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{4}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - x_m)^2, \tag{5}$$

$$CI = x_m \mp \frac{\sigma t}{\sqrt{n}}, \tag{6}$$

where $t$is the value of the t-distribution [19] , n is the number of observations, and σ is the standard deviation of the samples.

The effect of the threshold coefficient on the performance of the GA-s was also studied. The simulations of Gaussian and sigmoid circuit design were repeated for different threshold coefficients. It is well known that the performance of the GA-c gets worse as population size decreases, so it was expected that the GA-s would perform better than the GA-c in a small population. This assumption was also tested for Gaussian and sigmoid circuits by repeating the simulations for different population sizes.

4.1. Gaussian and sigmoid circuit

In order to illustrate the performance of the GA-s, the 99% CI of the Gaussian circuit for various threshold coefficients are plotted in Figure 5. The $\tau = 0$ case indicates the GA-c. It is observed in Figure 5 that the mean best fitness values of the GA-s are always lower than the mean values of the GA-c. Besides, the GA-s has better results with 99% confidence for $\tau \geq 0.5$. As the speciation threshold coefficient decreases, the performance of the GA-s also decreases. This observation contradicts the assertion in Section 2 about the quality of building blocks. However, an elaborate investigation reveals that, as expected, there is a trade-off between the threshold coefficient and the occurrence probability of speciation. This situation is illustrated in Figure 6.

In Figure 6, the percentage of the speciation number decreases drastically for $\tau < 0.4$. Evidently, the speciation algorithm never takes place in many simulations, and they have the same results as the GA-c. Therefore, the mean value and the standard deviation of the fitness value increase for $\tau < 0.4$. Although it is not given here, if the simulations resulting in a lack of speciation are eliminated, and only the results where speciation takes place are considered, the performance of the GA-s increases significantly.

Figure 7 shows the box-and-whisker plots [20] of the simulation results for the GA-c and GA-s versus τ. The outliers are ignored in the plot without losing any statistical information.

It is observed that the third quartile of the GA-s is almost equal to the first quartile of the GA-c. That means that approximately 75% of the simulation results of the GA-s are better than the results found by the

GA-c, and at least 50% of the simulation results of the GA-s have better fitness values than the corresponding GA-c simulations. Since the simulations were performed with the same seed values, it can be safely concluded that while the GA-c was stuck in a local minimum, the proposed GA-s can find better solutions in analog circuit design.

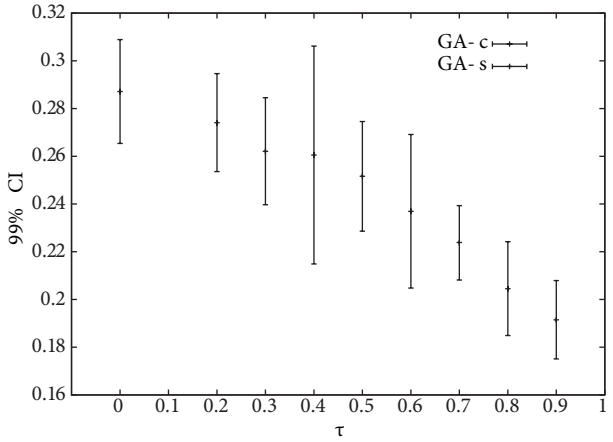

Figure 5. Speciation threshold coefficient versus 99% confidence intervals for the Gaussian circuit ($\tau = 0$ for GA-c, population size = 2000).

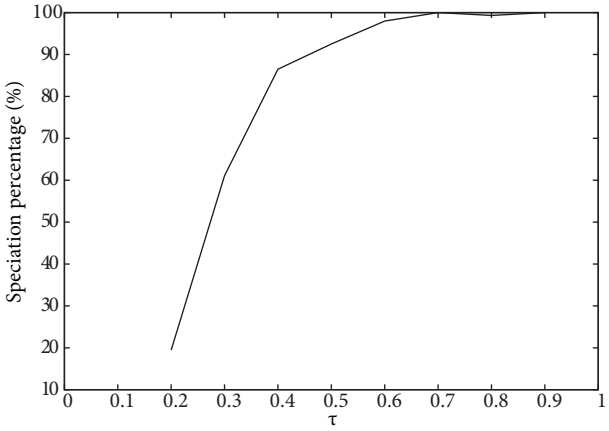

Figure 6. Threshold coefficient versus speciation occurrence percentage for the Gaussian circuit (population size = 2000).

The performance variation of the GA-s for different sizes of populations was evaluated for Gaussian and sigmoid circuit design problems. The Gaussian circuit design problem was solved 125 times for different population sizes with the GA-s and GA-c. The 99% CIs of these simulations are shown in Figure 8. The threshold coefficient of the speciation algorithm is chosen as 0.8 for these simulations.

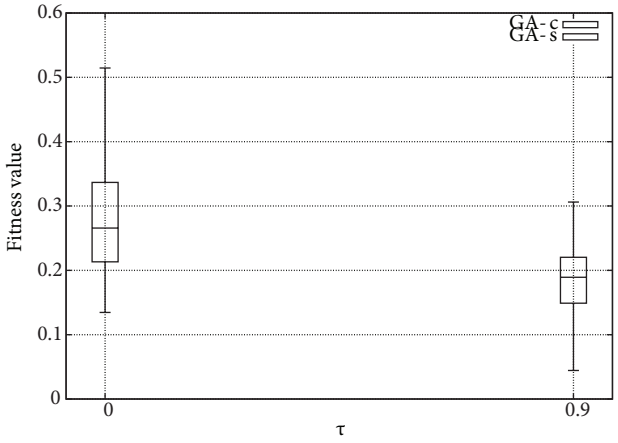

Figure 7. Box-and-whisker plot of the Gaussian circuit for GA-c and GA-s ($\tau = 0$ for GA-c, $\tau = 0.9$ for GA-s).

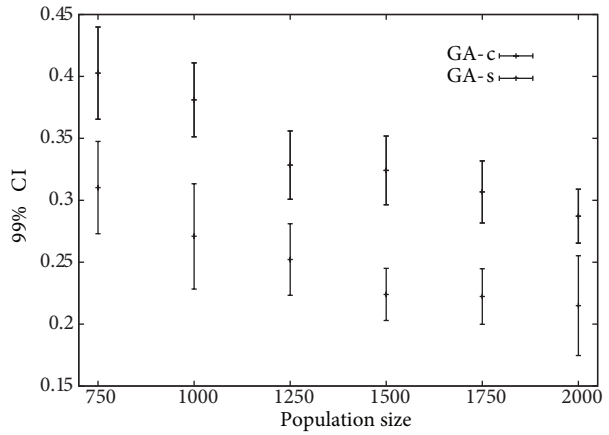

Figure 8. Population size versus 99% confidence intervals of the Gaussian circuit ($\tau = 0.8$).

In Figure 8, the lower CIs belong to the GA-s. Considering the CIs of the GA-s and GA-c at population sizes of 1000 and 2000, respectively, it can be concluded that the same solution quality can be obtained with the GA-s for almost half the size population. Since a large population demands more simulation time, analog

circuit design can be done in less time with the GA-s. The performance of the proposed method over the GA-c increases as the population size decreases. This tendency is clearly observed in sigmoid circuit design in Figure 9.

Figure 9 illustrates the 99% CIs of GA-s and GA-c versus population size. The threshold coefficient was 0.8 for these simulations. GA-s outperformed GA-c in each case. The gaps between CIs indicate that GA-s performance over GA-c increases as the population size decreases.

Sigmoid circuit design problem was evaluated for different threshold coefficients using the same procedures as the Gaussian circuit design. Figure 10 shows the CIs of the simulation results.

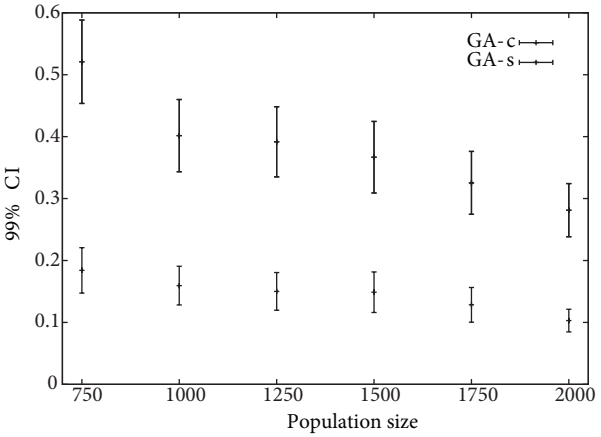

Figure 9. Population size versus 99% confidence intervals for the sigmoid circuit ($\tau = 0.8$).

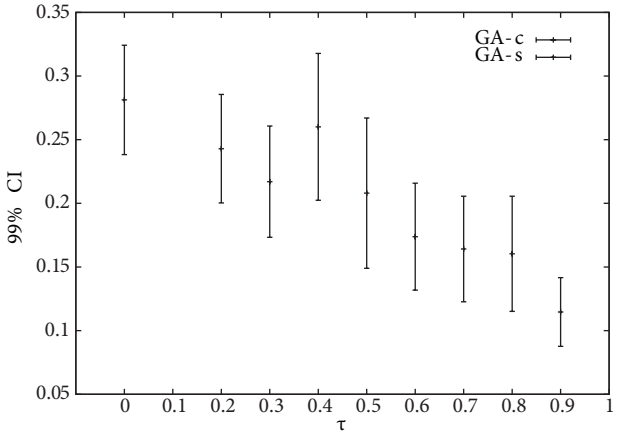

Figure 10. Threshold coefficient versus 99% confidence intervals for the sigmoid circuit ($\tau = 0$ for GA-c, population size = 2000).

Here, the same characteristics observed before were observed again. While the GA-s outperformed the GA-c for threshold coefficients higher than 0.6, its performance deteriorated for the lower value because of the occurrence rate of the speciation algorithm. This is shown in Figure 11.

The box-and-whisker plot graph for the sigmoid circuit is given in Figure 12.

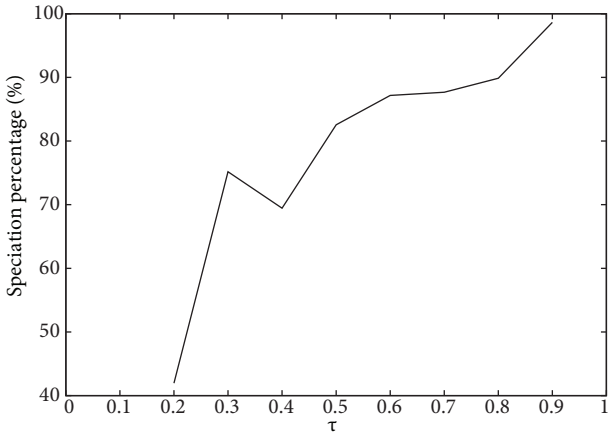

Figure 11. Threshold coefficient versus occurrence percentage of the speciation algorithm for the sigmoid circuit (population size = 2000).

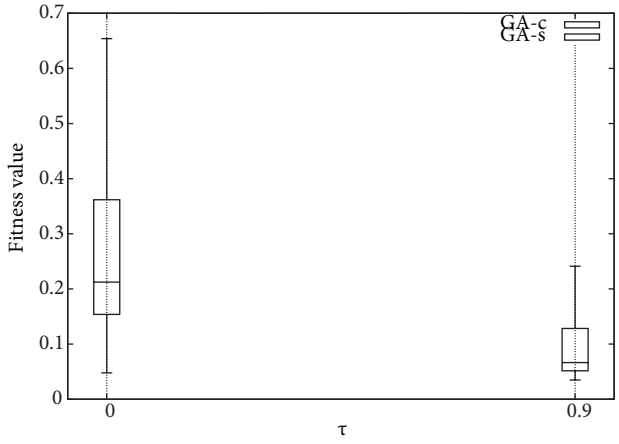

Figure 12. The box-and-whisker plot of the sigmoid circuit for GA-c and GA-s ($\tau = 0$ for GA-c, $\tau = 0.9$ for GA-s).

Clearly, the third quartile of the GA-s is lower than the first quartile of the GA-c. At least 50% of GA-s simulations find better solutions than the GA-c in the same conditions.

4.2. Square and cube circuit

The GA-s was also tested for square and cube circuits. The 99% CIs of the simulation results are given in Figure 13.

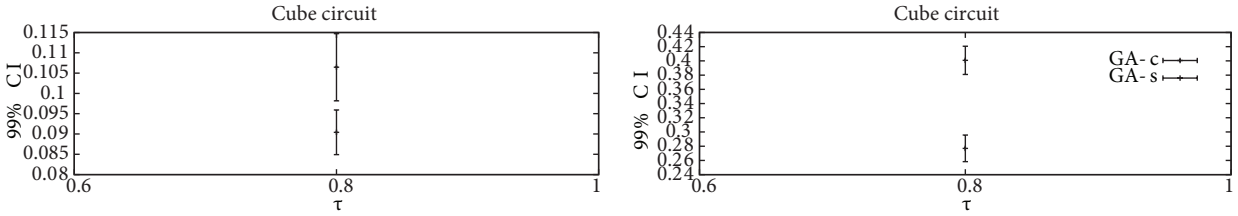

Figure 13. The 99% confidence interval plots for cube and square circuits (cube circuit population = 300, square circuit population = 2000).

Figure 13 indicates that the GA-s gives better results for two circuit examples. The results are consistent with the previous experimental results. Although it is not given here, the square and cube circuit designs were tested for different τ coefficients and the same pattern as in Gaussian and sigmoid circuit design was obtained.

Figure 14 shows box-and-whisker plots of the cube and square circuits.

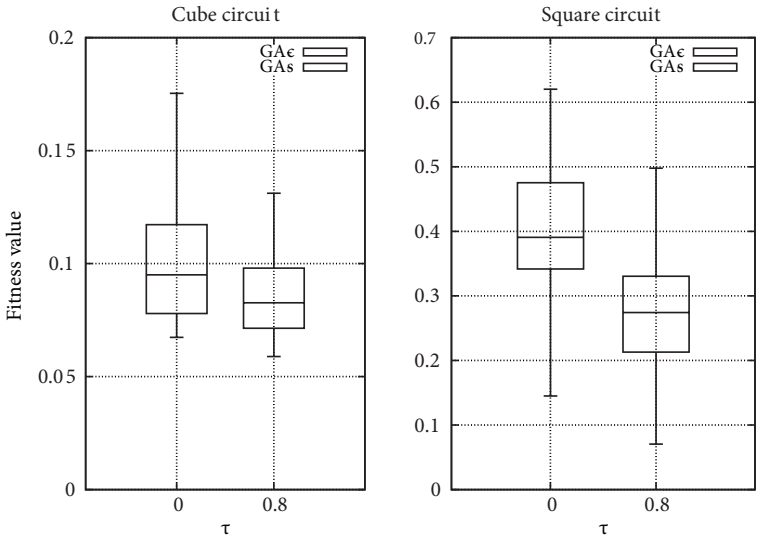

Figure 14. The box-and-whisker plot of the cube and square circuits for GA-c and GA-s ($\tau = 0$ for GA-c, $\tau = 0.8$ for GA-s).

The same pattern observed in the sigmoid and Gaussian circuits is also seen in Figure 14. The gap between the CIs of the square circuit is wider than the gap between the CIs of the cube circuit in Figure 13. Therefore, the performance of the GA-s over the GA-c in the square circuit problem was better than the performance of the GA-s in the cube circuit problem. This situation is observed in Figure 14.

5. Conclusion

This paper introduces a speciation-based local search procedure for GA in analog circuit design. The effect of speciation was tested in sigmoid, Gaussian, cube, and square circuit design problems. Statistical analyses were

carried out on problems to prove the effectiveness of the method. In all problems, the speciation algorithm produced better solutions with 99% confidence. It is obvious that the local search ability of the GA-c is insufficient, and it can easily miss good building blocks. The speciation algorithm improves the local search capability of the GA. The proposed algorithm does not require extra circuit simulations to perform local searches. Hence, the method is better than other local search methods in the literature in terms of computation time and computation power. It is also more statistically consistent regarding CIs.

Acknowledgment

The numerical calculations reported in this paper were performed at the TÜBİTAK ULAKBİM High Performance and Grid Computing Center (TRUBA Resources).

References

[1] Koza JR, Bennett FH 3rd, Andre D, Keane MA, Dunlap F. Automated synthesis of analog electrical circuits by means of genetic programming. IEEE T Evolut Comput 1997; 1: 109–128.

[2] Sripramong T, Toumazou C. The invention of CMOS amplifiers using genetic programming and current-flow analysis. IEEE T Comput Aid D 2002; 21: 1237–1252.

[3] Hu J, Zhong X, Goodman ED. Open-ended robust design of analog filters using genetic programming. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation; 25–29 June 2005; Washington, DC, USA. New York, NY, USA: ACM Press. pp. 1619–1626.

[4] Goh C, Li Y. GA automated design and synthesis of analog circuits with practical constraints. In: Proceedings of the Congress on Evolutionary Computation; 27–30 May 2001; Seoul, Republic of Korea. New York, NY, USA: IEEE. pp. 170–177 vol 1.

[5] Tlelo-Cuautle E, Guerra-Gomez I, Duarte-Villasenor MA, de la Fraga LG, Flores-Becerra G, Reyes-Salgado G, Reyes-Garcia CA, Rodriguez-Gomez G. Applications of evolutionary algorithms in the design automation of analog integrated circuits. J Appl Sci 2010; 10: 1859–1872.

[6] Rothlauf F. Representations for Genetic and Evolutionary Algorithms. 2nd ed. Berlin, Germany: Springer-Verlag, 2006.

[7] McConaghy T, Palmers P, Steyaert M, Gielen GGE. Trustworthy genetic programming-based synthesis of analog circuit topologies using hierarchical domain-specific building blocks. IEEE T Evolut Comput 2011; 15: 557–570.

[8] Razavi B. Design of Analog CMOS Integrated Circuits. New York, NY, USA: McGraw-Hill Inc., 2000.

[9] Sansen WMC. Analog Design Essentials. Berlin, Germany: Springer, 2006.

[10] Hwang KS, Cho SB. Improving evolvable hardware by applying the speciation technique. Appl Soft Comput 2009; 9: 254–263.

[11] Tazawa I, Koakutsu S, Hirata H. An immunity based genetic algorithm and its application to the VLSI floorplan design problem. In: Proceedings of International Conference on Evolutionary Computation; 20–22 May 1996; Nagoya, Japan. New York, NY, USA: IEEE. pp. 417–421.

[12] Chen J, Zhu W. A hybrid genetic algorithm for VLSI floor planning. In: Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS); 29–31 October 2010; Xiamen, China. New York, NY, USA: IEEE Vol. 2. pp. 128–132.

[13] Song X, Cui Y, Li A. Optimization algorithm of evolutionary design of circuits based on genetic algorithm. In: Proceedings of the 5th International Symposium on Computational Intelligence and Design (ISCID); 28–29 October 2012; Hangzhou, China. New York, NY, USA: IEEE. pp. 336–339.

[14] Tai Y, Liu Z, Yu H, Liu J. Efficiency optimization of induction motors using genetic algorithm and hybrid genetic algorithm. In: Proceedings of the 2011 International Conference on Electrical Machines and Systems (ICEMS); 20–23 August 2011; Beijing, China. New York, NY, USA: IEEE. pp. 1–4.

[15] Lo IH, Li Y, Li K. Hybrid genetic algorithm with mixed mutation mechanism for optimal display panel circuit design. In: Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI); 18–20 November 2010; Hsinchu City, Taiwan. New York, NY, USA: IEEE. pp. 222–225.

[16] Coello Coello CA, Alba E, Luque G. Comparing different serial and parallel heuristics to design combinational logic circuits. In: 5th NASA / DoD Workshop on Evolvable Hardware; 9–11 July 2003; Chicago, IL, USA. New York, NY, USA: IEEE. pp. 3–12.

[17] Damavandi N, Safavi-Naeini S. A hybrid evolutionary programming method for circuit optimization. IEEE T Circuits-I 2005; 52: 902–910.

[18] Huang T, Huang J, Zhang J. An orthogonal local search genetic algorithm for the design and optimization of power electronic circuits. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation; 1–6 June 2008; Hong Kong. New York, NY, USA: IEEE. pp. 2452–2459.

[19] Holman JP. Experimental Methods for Engineers. New York, NY, USA: McGraw-Hill Inc., 1994.

[20] McGill RT, John WL, Wayne A. Variations of box plots. T Am Stat 1978; 32: 12–16.