# GA and ACO in hybrid approach for Analog Circuit Performance Optimization

B. Benhala, A. Ahaitouf

Faculty of Sciences and Technology, University of Sidi Mohamed Ben Abdellah, Fes, Morocco

***Abstract*— There are several types of problems where a direct optimization algorithm could fail to obtain an optimal solution. This clearly paves way to the need for hybridization of meta-heuristics to improve the performance of the algorithm: speed of convergence or the quality of the solutions. In this paper we present two hybrid search algorithms combining the advantages of genetic algorithms (GA) and ant colony optimization (ACO) that can explore the search space and exploit the best solutions. The hybrid algorithms (GAACO or ACOGA) are detailed and their performances in terms of optimum quality and computing time are checked, via two applications that consist of optimizing performances of a CMOS second generation current conveyor (CCII), and an operational amplifier (Op-Amp).**

***Keywords—Genetic Algorithm, Ant Colony Optimization, Hybridation, Current Conveyor, Operational Amplifier, Analog Design.***

## I. Introduction

Analog Circuit Design is a difficult task due to the number of parameters characterizing such circuits. Due to this complexity, is used simple statistic-based methods to refine a sizing or to satisfy some specifications [1]. Nowadays, focal interest of analogue designers on the applications of meta-heuristics algorithms to the analogue circuit design [2]. Indeed, these methods can be applied to solve the sizing problems of complex designed systems/circuits whose aim is to find the optimal parameters' values (length and width of MOS transistors, bias current, capacitor values, etc.) in such a way that the final circuit performances meet as close as possible the design requirements, while satisfying a set of constraints. Main used such algorithms and techniques are Genetic Algorithms (GA) [3], Simulated Annealing (SA) [2], Swarm Optimization technique (PSO) [4] and Ant Colony Optimization technique (ACO) [5,6].

The meta-heuristic techniques should achieve an appropriate balance between the exploitation (intensification) of the search experience achievement and the exploration (diversification) areas of the search space unvisited or relatively unexplored [7,8]. The choice between the processes of diversification and intensification is a delicate problem in the design and operation of meta-heuristics. To overcome this problem, it would be interesting to use two different algorithms and assigning to each of operation diversification or intensification.

In our previous work [5,6,9], we have employed GA and ACO to solve analog circuit sizing problems. Even though those approaches could find the best solution in those simulated cases, the search efficiency seemed not good enough. In fact, the ACO is time consuming compared to GA and this last, gives optimal solutions relatively less good than those generated by the ACO.

The GA belongs to evolutionary Algorithms [10] and the ACO is part of the Swarm Intelligence techniques (SI) [11]. To take advantage of both techniques, we propose in this paper two hybrid meta-heuristics: Genetic Algorithm Ant Colony Optimization (GAACO) and Ant Colony Optimization Genetic Algorithm (ACOGA). Where each algorithm (GA or ACO) occupies either diversification or intensification and vice versa.

The hybrid algorithms (GAACO and ACOGA) are detailed and are evaluated through two application examples a second generation current conveyor (CCII) and a two-stage CMOS operational amplifier (Op-Amp). The aim is to compare the techniques (GA, ACO, GAACO and ACOGA) in terms of quality of the solutions and computing time.

This paper is structured as follows: in section II, we give a brief overview of the GA and ACO techniques. In section III we present pseudo codes of GAACO and ACOGA algorithms. In section IV and V, the optimization results obtained by each technique are presented and compared. SPICE simulation results are given to show viability of the obtained results. In section VI, we give some concluding remarks.

## II. A breif overview to GA and ACO

### *A. Genetic Algorithm*

Genetic algorithms (GAs) differ from the traditional approaches of existing optimization techniques. The simple ideas of the GA search have their roots in the biological processes of survival and adaptation.

The basic principle of genetic algorithms consists of sampling a population of potential solutions. A population of individuals is, initially, randomly generated. The GA performs then operations of selection, crossover and mutation on the individuals, corresponding respectively to the principals of survival of the fittest, recombination of genetic material and random mutation observed in nature [12]. The optimization process is carried out through the generation of successive populations until a stop criterion is met.

To implement the genetic algorithm technique, the following parameters need to be selected [13]:

- Population size (N),
- Probability of crossover (pcross) (between 0.7-1.0).
- Probability of mutation (pmut) (between 0.01-0.05)

The pseudo code of the GA procedure is as follows:

```
Random initialization of the population
      max_fitness := 0
Do
For each member chromosome
       fitness := Fitness_Evaluation (chomosome)
        If fitness > max_fitness
             max_fitness := fitness
             fittest_solution = chromosome
        End if
      End for
      While generation < max_generations
      offspring := Selection (parents)
      fitness := Fitness_Evaluation (offspring)
         If fitness > max_fitness
             max_fitness := fitness
             fittest_solution = offspring
         End if
save fittest_solution
END
```

ALGORITHM 1. PSEUDO CODE OF GA.

### B. *Ant Colony Optimization*

The first Ant Colony Optimization (ACO) called Ant System (AS) was inspired through studying of the behavior of ants in 1991 by Macro Dorigo and al. [14].

The ACO technique is inspired by the collective behavior of deposition and monitoring of some traces as it is observed in insect colonies [15,16], such as ants. It is for example well known that ants deposit pheromone on the ground in order to mark some favourable paths that should be followed by other members of the colony.

For solving such problems, ants randomly select the vertex to be visited. When an ant $k$ is in the vertex $i$, the probability for going to the vertex $j$ is given by the following expression [15, 17]:

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}}{\sum_{l \in J_i^k} (\tau_{il})^{\alpha}.(\eta_{il})^{\beta}} & if \quad j \in J_i^k \\ 0 & if \quad j \notin J_i^k \end{cases} \tag{1}$$

where $J_i^k$ is the set of neighbors of the vertex $i$ of the $k^{th}$ ant, $\tau_{ij}$ is the amount of pheromone trail on the edge ($i$, $j$), $\alpha$ and β are weightings that control the pheromone trail $\tau_{ij}$ and the visibility value, $\eta_{ij}$ given by:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{2}$$

Where $d_{ij}$ is the distance between vertices $i$ and $j$.

The pheromone rate values are updated during each iteration by all the m ants that have built a solution in the iteration itself. The pheromone rate $\tau_{ij}$, which is associated with the edge joining vertices $i$ and $j$, is updated as follows:

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{3}$$

where ρ is the evaporation rate, $m$ is the number of ants, and $\Delta\tau_{ij}^{k}(t)$ is the quantity of pheromone laid 'deposited, or dropped of'on edge ($i$, $j$) by ant $k$:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L^k} & \text{if ant k used edge (i, j) in its tour,} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$Q$ is a constant and $L_k$ is the length of the tour constructed by the ant $k$.

The pseudo code of the ACO procedure is as follows:

```
Random initialization of the pheromone value
Do
   For each iteration
      For each ant
            Compute of the probability P according to (1)
            Determine the Pmax
          End
         Compute OF
         End
      Deduce the best OF
      Update pheromone values according to (3)
      End
   Report the best solution
END
```

ALGORITHM 2. PSEUDO CODE OF ACO.

## III. THE HYBRID GAACO /ACOGA

The main idea in this paper (see Figure 1) is to merge the two meta-heuristics (GA and ACO). The question here is: which algorithm we will start and when the second enters to the hybridization process?

So we'll propose two hybrid algorithms: GAACO and ACOGA used for the optimal sizing of different aspect analogue circuits.

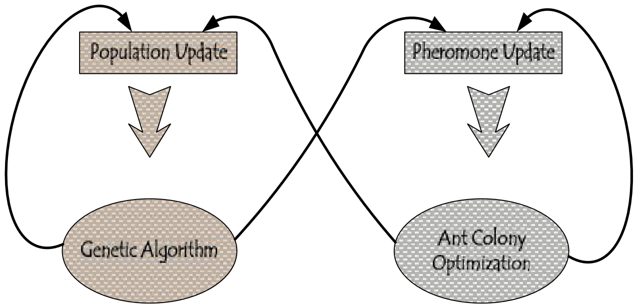


Fig. 1. Hybridation strategy.

The procedure of GAACO algorithm is to apply initialization, probabilistic search, pheromones update and GA for the shortest route. These above procedures are iterated until the termination criterion is satisfied. The pseudo code of the GAACO procedure is as follows:

Random initialization of the pheromone value
**Do**
  **For** each iteration
    **Procedure of the ACO algorithm**
  **End**
  Initialization of the population according to the best solutions from ACO algorithm
  **For** each iteration
    **Procedure of the GA algorithm**
  **End**
 Report the best solution
**END**

ALGORITHM 3. PSEUDO CODE OF GAACO.

The procedure of ACOGA algorithm is to apply initialization, selection, crossover, mutation, fitness evaluation, and ACO for the shortest route. These above procedures are iterated until the termination criterion is satisfied. The pseudo code of the ACOGA procedure is as follows:

Random initialization of the population
**Do**
  **For** each iteration
    **Procedure of the GA algorithm**
  **End**
  Report the best solutions
Initialization of the pheromone value according to the best solutions from GA algorithm
  **For** each iteration
    **Procedure of the ACO algorithm**
  **End**
Report the best solution
**END**

ALGORITHM 4. PSEUDO CODE OF ACOGA.

## IV. OPTIMIZING PERFORMANCES OF CCII+

In this section we applied the four algorithms (GA, ACO, GAACO and ACOGA) to perform optimization of CCII+ (Figure 2) including constraints like saturation conditions. The optimization techniques work on MATLAB codes and are able to link SPICE to measure performances.

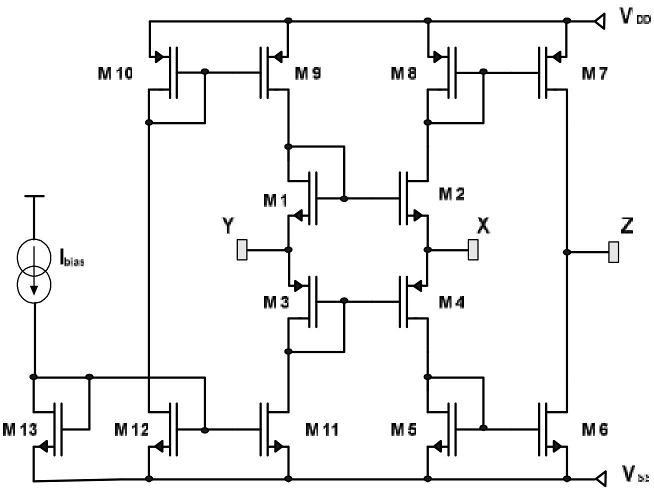


Fig. 2. A second generation current conveyor (CCII+).

As already mentioned above, the objective functions to be optimized are:

- $R_X$: the X-port input parasitic resistance to be minimized.
- *fci*: the current high cut off frequency to be maximized.

The geometric variables that will be used to optimize performances of a CMOS positive second generation current conveyor (CCII+) are the MOS transistors sizes: channels lengths (LN, and LP) and gates widths (WN and WP) while respecting the saturation conditions of the transistors MOS. The studied algorithms parameters are given in Table 1 with a generation algorithm of 1000.

TABLE I. THE ALGORITHM PARAMETERS

| | | |
|---|---|---|
| **ACO** | Number of Ants | 100 |
| | Evaporation rate (ρ) | 0.1 |
| | Quantity of deposit pheromone (Q) | 0.2 |
| | Pheromone Factor (α) | 1 |
| | Heuristics Factor (β) | 1 |
| **GA** | Population size | 100 |
| | Crossover Probability | 0.9 |
| | Mutation Probability | 1e-4 |

The ACOGA and GAACO algorithm parameters are those in Table 2 except that the number of ants is 50 for ACO and the number of the population is 50 for GA.

Figures 3 and 4 show a comparison between results for *fci* and *Rx* respectively, obtained using GA, ACO, ACOGA and GAACO, where the rapid convergence of the hybrid algorithms can be noticed.

Figure 3 shows the *fci* values versus to the number of generation.

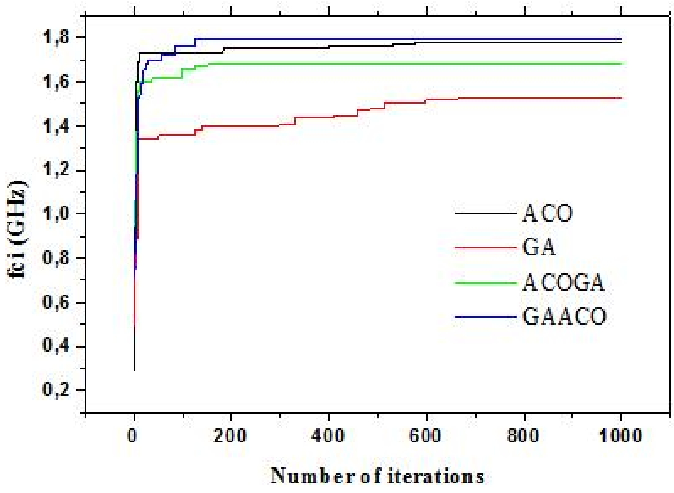


Fig. 3. The *fci* values *vs.* number of iterations.

Figure 4 shows the *Rx* values versus to the number of generation.

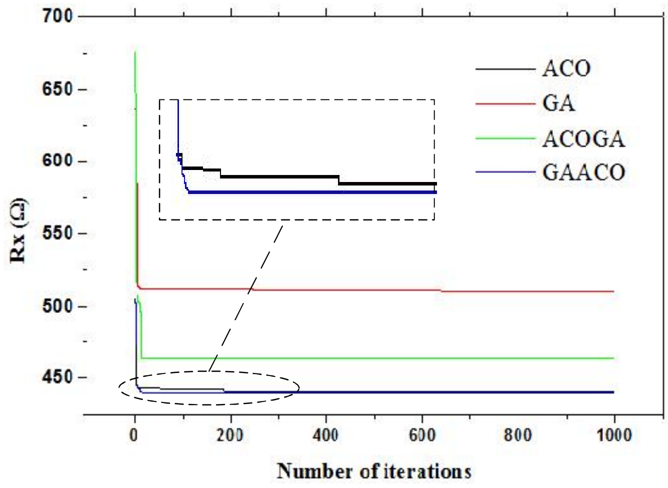


Fig. 4. The *Rx* values *vs.* number of iterations.

The optimum sizes obtained after optimization are simulated with SPICE to measure the performances. The optimization and simulation results are collected in the tables II and III respectively for *fci* and *Rx*. SPICE simulations are performed using the technology of 0.35 µm CMOS from AMS. The supply voltages used are -2.5V/2.5 V and a bias current Io = 100 µA.

TABLE II. OPTIMIZATION AND SIMULATION RESULTS FOR FCI-MAX

| | Optimum size (µm) | | | | *fci max* (GHz) | |
|---|---|---|---|---|---|---|
| | LN | WN | LP | WP | **Optimal** | Simulated |
| **GA** | 0.60 | 5.98 | 0.36 | 10.67 | **1.541** | 1.546 |
| **ACO** | 0.55 | 5.10 | 0.35 | 08.91 | **1.792** | 1.787 |
| **GAACO** | 0.58 | 5.39 | 0.35 | 09.79 | **1.688** | 1.683 |
| **ACOGA** | 0.60 | 4.53 | 0.35 | 07.74 | **1.802** | 1.793 |

TABLE III. OPTIMIZATION AND SIMULATION RESULTS FOR RX-MIN

| | Optimum size (µm) | | | | *Rx min* (Ω) | |
|---|---|---|---|---|---|---|
| | LN | WN | LP | WP | **Optimal** | Simulated |
| **GA** | 0.56 | 15.97 | 0.39 | 28.24 | **510** | 518 |
| **ACO** | 0.55 | 20.36 | 0.36 | 30.00 | **441** | 447 |
| **GAACO** | 0.59 | 19.20 | 0.36 | 30.00 | **463** | 472 |
| **ACOGA** | 0.55 | 19.28 | 0.35 | 30.00 | **439** | 451 |

In order to check the convergence rate of the proposed algorithms, a robustness test was performed. *i.e.* the algorithms are applied a hundred times for optimizing the *Rx* and *fci* objectives. In Figure 5 and 6 we present obtained results (respectively for *fci* and *Rx*) for the algorithms: GA, ACO, GAACO and ACOGA.

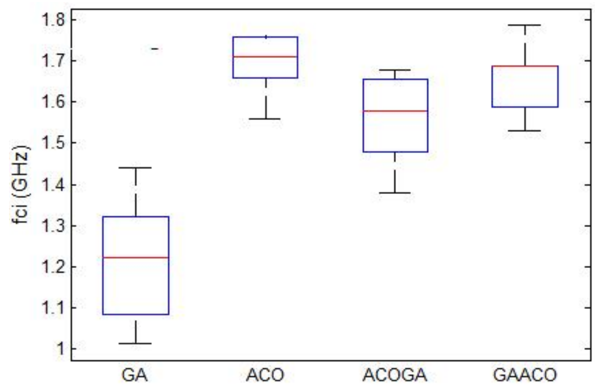


Fig. 5. Results obtained for 100 generations of the four algorithms for *fci*.

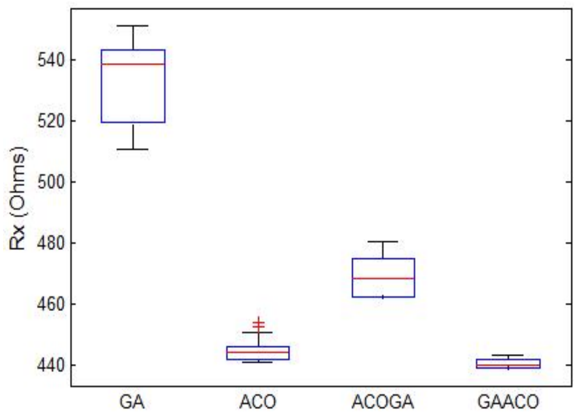


Fig. 6. Results obtained for 100 generations of the four algorithms for *Rx*.

The good convergence ratio can be easily noticed, despite the probabilistic aspect of the four algorithms. We can, also, notice that the robustness of the ACO and GAACO algorithms is better than the robustness of the GA and ACOGA algorithm; in fact the convergence rate to the same optimal value are 12%,47%, 23% and 51% respectively for GA, ACO, ACOGA and GAACO.

In Table 4 we present a comparison between computing times of the optimization algorithms, both performed with 1000 generations.

The four Matlab algorithms (GA, ACO, ACOGA and GAACO) was implemented in Intel Core 2 Duo CPU T5800 @ 2.00GHZ 2.00GHZ.

TABLE IV. COMPARISON OF THE COMPUTING TIME OF THE FOUR ALGORITHMS

| | Running Time (seconds) | | | |
|---|---|---|---|---|
| | **GA** | **ACO** | **ACOGA** | **GAACO** |
| *Rx min* | 24.20 | 113.38 | 72.19 | 74.30 |
| *fci max* | 30.18 | 124.15 | 83.52 | 84.05 |

We notice that the GA algorithm is faster than the other algorithms and the execution time is almost the same for both hybrid algorithms.

## V. APPLICATION OF HYBRIDATION TO THE OP-AMP

The schematic of two stage CMOS operational amplifier (Op-Amp) is shown in Figure 7.

Performances of the Op-Amp are evaluated via several parameters:

- Open-loop voltage gain Av
- Common Mode Rejection Ratio CMRR
- Die Area A
- Power dissipation P

The Av and CMRR to be maximized; A and P to be minimized.

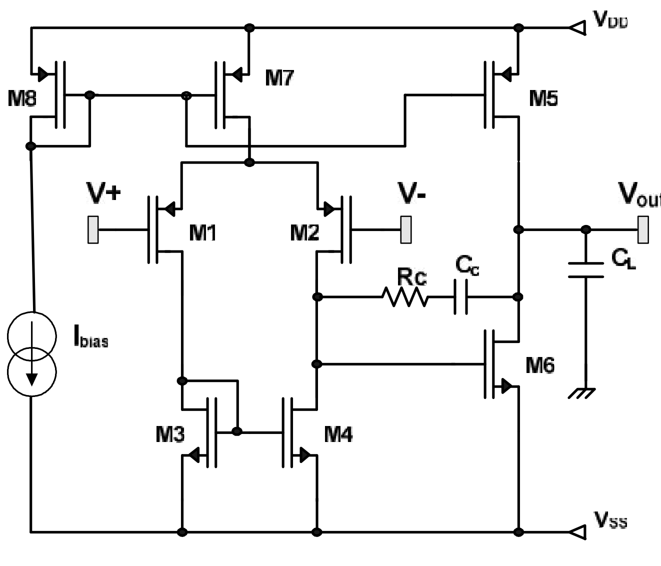


Fig. 7. A two stage CMOS operational amplifier (Op-Amp).

The follow parameters are considered fixed, e.g., the compensation resistor ($R_C$=600Ω), the compensation capacitor ($C_C$=2pF) and the capacitive load ($C_L$=10pF).

The four algorithms was applied to optimize the MOS transistors sizes: $W_1$,$W_2$, $W_3$,$W_5$,$W_6$,$W_7$,$W_8$, and the value of the bias Io.

Table 5 shows the optimum size of the Op-Amp. The optimal performances and simulation results are collected in the tables 6.

TABLE V. OPTIMIZATION RESULTS

| | $W_{1,2}$ (µm) | $W_{3,4}$ (µm) | $W_5$ (µm) | $W_6$ (µm) | $W_7$ (µm) | $W_8$ (µm) | L (µm) | Io (µA) |
|---|---|---|---|---|---|---|---|---|
| **GA** | 212.30 | 167.94 | 51.00 | 567.74 | 86.21 | 9.18 | 0.42 | 15.70 |
| **ACO** | 210.00 | 185.67 | 57.00 | 434.10 | 82.00 | 9.71 | 0.37 | 10.00 |
| **ACOGA** | 211.40 | 194.39 | 53.91 | 497.39 | 68.97 | 8.88 | 0.38 | 10.80 |
| **GAACO** | 210.00 | 192.28 | 57.92 | 437.88 | 65.94 | 9.66 | 0.35 | 10.00 |

TABLE VI. PERFORMANCE AND SIMULATION RESULTS

| | | Av (dB) | CMRR (dB) | A ($\mu m^2$) | P (mW) |
|---|---|---|---|---|---|
| **GA** | **Optimal** | **115.40** | **112.90** | **612** | **5.01** |
| | Simulated | 111.26 | 107.43 | --- | 5.23 |
| **ACO** | **Optimal** | **120.16** | **117.66** | **555** | **3.06** |
| | Simulated | 115.48 | 111.63 | --- | 3.12 |
| **ACOGA** | **Optimal** | **119.38** | **116.88** | **541** | **3.21** |
| | Simulated | 113.94 | 110.28 | --- | 3.33 |
| **GAACO** | **Optimal** | **120.76** | **118.26** | **482** | **2.76** |
| | Simulated | 115.57 | 111.96 | --- | 3.02 |

In Table 7 we present a comparison between computing times of the optimization algorithms.

TABLE VII. COMPARISON OF THE COMPUTING TIME OF THE FOUR ALGORITHMS

| **Algorithms** | **Running Time (seconds)** |
|---|---|
| GA | 052.37 |
| ACO | 491.52 |
| ACOGA | 238.13 |
| GAACO | 267.04 |

One can clearly notice that the GA algorithm is always faster than the other algorithms and the execution time is almost the same for both hybrid algorithms.

Figure 7 presents a Radio Chart representation of the performance optimality of each algorithm. It can easily conclude that GAACO results are the better ones.

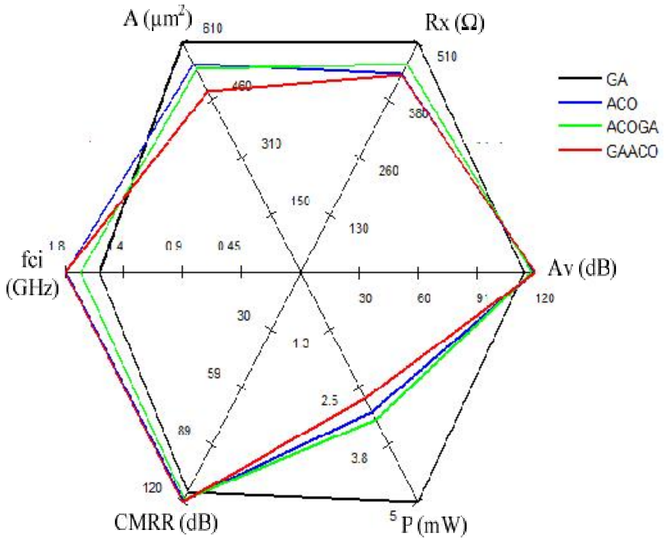


Fig. 7. Performance optimality for the four algorithms.

The following table resumes and compares the main features of the GA, ACO, ACOGA and GAACO algorithms.

TABLE VIII. COMPARISON OF THE PERFORMANCES OF THE FOUR ALGORITHMS

| **Algorithms** | **Running Time** | **robustness** | **Optimality** |
|---|---|---|---|
| **GA** | Good | Bad | Bad |
| **ACO** | Bad | Good | Good |
| **ACOGA** | Medium | Medium | Medium |
| **GAACO** | Medium | Good | Good |

## VI. CONCLUSIONS

The presented work proposes hybridization of two meta-heuristic techniques; i.e. Genetic Algorithm and Ant Colony Optimization; for dealing with the optimal sizing of analog circuits. These two hybrid algorithms are applied to optimize performances of two analog circuits: the CMOS second generation current conveyor and an operational amplifier. Optimal parameters (transistors' widths and lengths), obtained thanks to the GA, ACO, ACOGA and ACOGA algorithms were used to simulate the two CMOS circuit. Viability of the techniques was proved via SPICE simulations. The optimization results show that the GAACO algorithm offers better results in terms of objectives and robustness than the ACOGA technique. Which allows us to conclude that the GA is more adapted in diversification while ACO is better in intensification. We can also conclude that the GAACO offer, more rapidly, the same optimal values as the ACO technique.

## REFERENCES


[1] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah, "Optimization of custom MOS circuits by transistor sizing," ICCAD96, San Jose, pp. 174-190, Nov 10-14, 1996.

[2] J. Dréo, A. Pétrowski, P. Siarry and E. Taillard, "Metaheuristics for Hard Optimization: Methods and Case Studies," Springer, 2006.

[3] J. B. Grimbleby, "Automatic analogue circuit synthesis using genetic algorithms," IEEE Proceedings-Circuits, Devices and Systems, 147(6), pp. 319-323, 2000.

[4] M. Fakhfakh, Y. Cooren, A. Sallem, M. Loulou and P. Siarry, "Analog Circuit Design Optimization through the Particle Swarm Optimization Technique," Journal of Analog Integrated Circuits & Signal Processing. 63(1), Springer , pp. 71-82, 2010.

[5] B. Benhala, A. Ahaitouf, M. Kotti, M. Fakhfakh, B. Benlahbib, A. Mecheqrane, M. Loulou, F. Abdi, and E. Barkane, "Application of the ACO Technique to the Optimization of Analog Circuit Performances," a chapter in Analog Circuits: Applications, Design and Performance, (Ed.) T. Cuautle, NOVA Science Publishers, 2011, pp. 235-255.

[6] B. Benhala, A. Ahaitouf, A. Mechaqrane, B. Benlahbib, F. Abdi, E. Abarkan and M. Fakhfakh, "Sizing of current conveyors by means of an ant colony optimization technique," Proceedings of the 2nd International Conference on Multimedia Computing and Systems (ICMCS'11), pp. 899–904, IEEE Press, 2011.

[7] M. Dorigo, and T. Stützle, "The ant colony optimization metaheuristic: Algorithms, applications and advances," In Handbook of Metaheuristics. Kluwer Academic Publishers, 2002, pp. 251–285.

[8] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, "Metaheuristics for Hard Optimization," Éditions Eyrolles, ISBN-10: 2212113684, 2003.

[9] A. Sallem, E. Bradai, M. Kotti, E. Gaddour, M. Fakhfakh, and M. Loulou, "Optimizing CMOS current conveyors through meta heuristics,"," 4th International Symposium on Computational Intelligence and Intelligent Informatics, ISCIII '09, pp. 167–171, 2009.

[10] T. Bäck, "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms," ISBN 0-19-509971-0, Oxford University Press Oxford, UK ©1996

[11] J. Kennedy, R.C. Eberhart, and Y. Shi, "Swarm Intelligence," Morgan Kaufmann Academic Press, 2001.

[12] R.L. Haupt and S.E.Haupt, "Practical Genetic Algorithms," (book) John Wiley & Sons, ISBN 0-471-45565-2, 2004.

[13] D. E. Goldberg, and J. R. Koza, "Genetic algorithms in search, optimization and machine learning," Workshop Notes, Computer Science Department, Stanford University, August 6-10, 1990.

[14] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive Feedback as a Search Strategy," Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy,1991.

[15] M. Dorigo, G. DiCaro and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life Journal, 5, pp. 137–172, 1999.

[16] M. Dorigo and S. Krzysztof, "An Introduction to Ant Colony Optimization," a chapter in Approximation Algorithms and Metaheuristics, a book edited by T. F. Gonzalez, 2006.

[17] M. Dorigo, B. Birattari, and T. Stuzle, "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique," IEEE Computational Intelligence Magazine, 1(4), pp. 28-39, 2006.