A Multi-Objective Ant Colony Optimization for Routing in Printed Circuit Boards

Heitor T. Azambuja[1], Nadia Nedjah[1], Luiza M. Mourelle[2]

[1]*Dept. de Engenharia Eletrônica e Telecomunicações, Universidade do Estado do Rio de Janeiro*
Rua São Francisco Xavier n° 524, 20550-900, Maracanã, Rio de Janeiro, Brazil
heitortazamba@gmail.com, nadia@eng.uerj.br
[2]*Dept. de Engenharia de Sistemas e Computação, Universidade do Estado do Rio de Janeiro*
Rua São Francisco Xavier n° 524, 20550-900, Maracanã, Rio de Janeiro, Brazil
ldmm@eng.uerj.br

Abstract. Automatic routing for Very Large-Scale Integration (VLSI) and Printed Circuit Board (PCB) design is crucial for optimizing the work of engineers. Provided by popular Computer-Aided Design (CAD) software, this tool is part of a long-standing research field. As electronics evolve, with components becoming smaller and faster, circuit design constraints grow more complex, presenting new challenges. To deal with these difficulties, strategies using computational intelligence algorithms are employed to provide viable routing solutions in a timely manner. Among these strategies, multi-agent and swarm algorithms are highly relevant, being frequently applied alongside other approaches or by themselves. This paper introduces a variation of the Ant Colony Optimization (ACO) algorithm for PCB routing, emphasizing length matching between traces. The optimization function simultaneously minimizes three objectives: trace length, the number of times that traces cross with each other, and the length difference between traces. In other words, the ants try to find the shortest possible trace that has the same length as the other traces without crossing with them. We tested our algorithm in five fundamental scenarios and conducted a statistical analysis of multiple executions for each. The results demonstrate that our algorithm is a viable approach for PCB trace routing with length matching.

Keywords: Ant Colony Optimization; Printed Circuit Boards; Automatic Routing.

1 Introduction

Printed Circuit Boards (PCBs) serve as the foundation for securing and connecting electronic components. Found in nearly all electronic devices, PCBs consist of laminated layers of alternating conductive and insulating materials, typically copper and fiberglass or phenolic. The conductive layers feature etched connection traces and component mounting planes, forming electronic circuits similar to wires on a flat surface. For more complex circuits, multiple layers of conductors are used.

Initially, these conductive layers start as continuous copper planes, with the circuit design etched by removing copper from areas where connections are not needed. The design is printed on the board, exposing only the copper regions that need to be removed. The board is then immersed in a corrosive solution that eliminates the unwanted copper, leaving behind the desired traces and planes of the printed design.

Previously, circuit design was done manually. However, advancements in electrical and electronic engineering have reduced the size of components and traces, and increased the importance of electromagnetic tolerances and limitations. Nowadays, computer-aided design (CAD) programs, specifically developed for electronic circuit design, are essential. These programs not only facilitate the drawing of intricate traces that are impossible to create by hand but also verify the electromagnetic tolerances of all circuit traces, alerting users to problematic areas. For complex circuits, this tolerance verification would be extremely laborious, if not impossible, without such programs.

As shown by Zhang et al. [1], high-frequency circuits face various challenges due to the positioning of components and geometric characteristics of the PCB traces. These challenges include reflection, interference (crosstalk), attenuation, delay, instability (jitter), and noise. To mitigate and avoid these issues, various design techniques must be employed. Traces must adhere to strict limitations regarding distance, thickness, maximum and minimum lengths, and the number of layers.

Particularly for data and clock traces in high-frequency circuits, it is crucial to ensure they are of equal length to prevent phase deviation and signal delay, and to minimize the effects of capacitance and parasitic inductance. These problems arise when a signal travels a greater distance than others, arriving at the destination at different times and compromising signal integrity, Anand et al. [2]. Due to spatial constraints and the positioning of components on the PCB, these traces cannot be straight. The longest trace is identified, and curves are added to the shorter ones so that all traces are of equal length, as shown in Figure 1. The dimensions of these curves must be carefully specified, as they can introduce problems instead of mitigating them, Alexander Weiler and Verma [3].

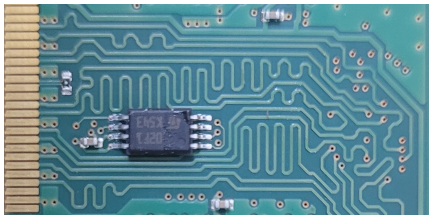


Figure 1. Traces with curves for length matching on a notebook memory module.

Although there are tools capable of automatically routing PCB traces, the process of length matching is mostly done manually. It requires the evaluation and intervention of the designer in each case to ensure the correct operation of the circuit.

Section 2 reviews related works utilizing swarm and multi-agent algorithms to address PCB and VLSI routing challenges. Section 3 delves into the canonical ACO algorithm. Section 4 explains our innovative technique, which adapts the ACO algorithm for PCB trace routing with an emphasis on length matching. Section 5 describes the test scenarios and presents the experimental results. Finally, Section 6 offers conclusions and closing remarks.

2 Related Works

During automatic routing, various scenarios can arise depending on the circuit's electrical characteristics and complexity. Key scenarios include connecting components, placing components and sizing the PCB, distributing clock signal traces to minimize delays, and matching trace lengths to reduce interference and electromagnetic issues. Both deterministic and computational intelligence techniques, often multi-agent algorithms, are used to address these scenarios.

In Nath et al. [4], a discrete PSO with genetic algorithm characteristics, called DPSO-MU, includes a mutation step. This step randomly selects particles and changes their positions to higher fitness values, constructed from the highest fitness particles in previous generations. This improves convergence rate and stability over the standard discrete PSO.

In Berlier [5], parallel genetic algorithms are used for component placement and routing order optimization to minimize trace length. Utilizing cloud computing resources, a 64-core machine runs multiple genetic algorithm populations simultaneously. This parallel strategy is 50 times faster than sequential execution and is effective for real circuit designs.

In Badriyah et al. [6], a genetic algorithm determines PCB component placement, minimizing PCB size, preventing overlap, and distributing high-heat components. Although this algorithm is slow and memory-intensive, it achieves satisfactory results for low-complexity circuits.

In Pandiaraj et al. [7], an algorithm based on Differential Evolution minimizes trace length in VLSI circuits. The algorithm partitions and overlaps circuit layers to maintain functionality while minimizing trace length, using trace length and layer count in the fitness function.

In Bhargab Sinha and Baishnab [8], a hybrid model combining Ant Colony Optimization with River Formation Dynamics (RFD) addresses VLSI routing challenges. RFD is a computational algorithm inspired by the natural process of river formation. It mimics how water naturally finds the lowest path and forms rivers, eroding the terrain and creating a path of least resistance over time. This model outperforms comparable techniques but requires more memory and processing power.

Regarding trace routing and geometry, related works using swarm intelligence and agent-based approaches typically optimize a single characteristic of the traces or perform multiple steps for each attribute. In contrast, our approach optimizes multiple characteristics simultaneously. Additionally, we apply the ACO algorithm, which is frequently used in routing problems but underutilized in PCB trace routing.

3 Ant Colony Optimization

Ant Colony Optimization, inspired by ant foraging behavior, is a swarm intelligence algorithm designed for population based search. Ants explore the search space guided by a heuristic function and pheromone trails, which indicate solution quality. During each iteration, ants build incremental solutions by depositing pheromones on paths, reinforcing better solutions over time. Ants are influenced by pheromone concentration and the best-known solution, leading them to explore high-pheromone areas [9].

Initially tested on the Travelling Salesman Problem (TSP), an NP-hard combinatorial optimization problem, ACO aims to find the shortest route visiting each city once and returning to the starting city. For n cities, there are $n!$ potential routes, with the goal being to minimize total travel distance. The probability of an ant k at position i choosing to move to position j is given by:

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in \mathcal{N}_i^k} (\tau_{il})^\alpha \cdot (\eta_{il})^\beta}, \quad j \in \mathcal{N}_i^k, \tag{1}$$

where τ_{ij} is the pheromone level, η_{ij} is the heuristic attractiveness (inverse distance for TSP), and α and β determine the influence of pheromone and heuristic information, respectively. $\mathcal{N}_i^k$ is the neighborhood of possible moves for ant k at position i. The pheromone trail update is:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \tag{2}$$

where ρ is the pheromone evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^k$ is the pheromone deposited by ant k, based on the quality of its solution (inverse path length for TSP).

4 PCB Trace Router with ACO

The ACO PCB trace router optimizes three objectives simultaneously: the trace length, the number of crossings with other traces, and the length matching between traces. The optimization function f_{opt} calculates the weighted average of these three objectives. In ACO, a colony is responsible for routing a Trace. For an ant k from colony c, the objective function that evaluates the trace is:

$$f_{opt}(k^{(c)}) = \frac{\omega_1 \Lambda_k^{(c)} + \omega_2 \Upsilon_k^{(c)} + \omega_3 \Gamma_k^{(c)}}{\omega_1 + \omega_2 + \omega_3}, \tag{3}$$

where: $\Lambda_k^{(c)}$ is the trace length, $\Upsilon_k^{(c)}$ is the number of times this trace crosses other traces, and $\Gamma_k^{(c)}$ is the length matching calculated by the sum of the length differences between this trace and other traces in the solution. ω_1, ω_2, and ω_3 are weights that regulate the importance of each objective.

The router's search space represents the copper surface of a rectangular PCB. This PCB area is divided into equally sized square cells. The PCB dimensions are specified in the algorithm as rows and columns, where a board with m rows and n columns contains $m \times n$ cells. Rows are numbered from bottom to top, and columns from left to right. Each cell can be visited by an ACO ant, with positions numbered starting from the bottom-left corner. Position 1 is the cell in the first row and first column, while position $n+1$ is directly above it in the second row and first column. Position i is located at (m_i, n_i).

ACO ants explore the cells, depositing pheromones to find the optimal trace path. A circuit consists of several traces connecting components, with each trace routed by a separate colony. Thus, the number of colonies in the ACO router is equal to the number of traces on the PCB, denoted by $\mathbb{C}$. Each colony has the same number of ants, represented by $\mathbb{K}$. Therefore, there are $\mathbb{C} \times \mathbb{K}$ ants in total. Each trace has predefined start and end positions.

Figure 2 illustrates the search space of a PCB with dimensions 6×17 cells, featuring 3 traces. Black dots denote cells that can be visited by any ant from any colony. Green dots mark the starting positions of the traces, while red dots signify the ending positions. Colonies are initialized at the green dots, and ants conduct the search to find the optimal path to the red dots.

The path constructed by an ant k from colony c is denoted as $\lambda_k^{(c)}$, and its length as $\Lambda_k^{(c)}$. This path is the sequence of positions the ant chooses to travel from the start position to the end position of its colony. An ant k from colony c at position i can move to any position in its neighborhood, as long as it is not the starting or ending

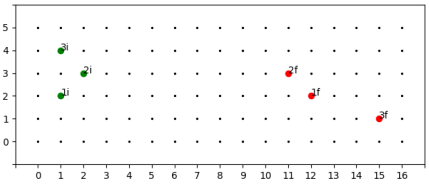


Figure 2. Representation of the search space of a PCB with 3 traces and 102 cells.

position of another colony. The neighborhood of position i, represented by $\mathcal{N}_i$, includes the cells directly to the right, left, above, and below i. The probability of an ant k belonging to colony c at position i moving to position j is given by:

$$p_{ij}^{(k_c)} = \frac{\left(\tau_j^{(c)}\right)^{\alpha} \cdot \left(\eta_j^{(c)}\right)^{\beta}}{\sum_{l \in \mathcal{N}_i} \left(\tau_l^{(c)}\right)^{\alpha} \cdot \left(\eta_l^{(c)}\right)^{\beta}}, \quad j \in \mathcal{N}_i, \tag{4}$$

here, $\tau_j^{(c)}$ is the pheromone of colony c associated with position j, $\eta_j^{(c)}$ is the heuristic information for colony c at position j, α and β are constants to determine the influence of pheromone and heuristic information, and $\mathcal{N}_i$ is the feasible neighborhood for position i. The heuristic information represents the attractiveness of an ant to a position. In the canonical ACO, this information is calculated by the inverse of the distance between the two positions, meaning the ant is more strongly attracted to a position with a shorter distance. Since in the ACO router, the search space is divided into square cells of equal dimensions, and the feasible neighborhood consists only of cells directly to the sides, above, and below the current position, the distance is the same for the entire neighborhood. Thus, only the pheromone would influence the choice of the next position. In this case, we opted to utilize the inverse of the pheromone of the other colonies in the position j. This approach causes the ant to be attracted to positions less used by ants from other colonies, avoiding "congested" regions. Additionally, it encourages the emergence of solutions without crossings with other paths. In this case, the heuristic information for colony c at position j is calculated by Eqution (5). The pheromone update is calculated by Equation (2), where the amount of deposited pheromone for ant k of colony c is given by Equation (6).

$$\eta_j^{(c)} = \left(\sum_{u \in \mathbb{C}} \tau_{ij}^{(u)} \right)^{-1}, \; u \neq c. \tag{5}$$

$$\Delta\tau_j^{k^{(c)}} = \frac{Q}{f_{opt}(k^{(c)})}, \tag{6}$$

where Q is a constant that regulates the amount of pheromone deposited. The optimization function $f_{opt}(k^{(c)})$ evaluates the path length $\Lambda_k^{(c)}$, the number of crossings with other paths $\Upsilon_k^{(c)}$, and the length matching $\Gamma_k^{(c)}$. The optimization function is in the denominator because these three objectives are minimized. $\Lambda_k^{(c)}$ is the number of steps in the path. The parameter $\Upsilon_k^{(c)}$ quantifies how many times the path constructed by ant k from colony c uses a position that is also used by the best path constructed by the other colonies, calculated by Equation (7). It counts how many times this path crosses with the other paths, considering only the best results found so far by the other colonies. $f_c(u, j)$ is defined by Equation (8).

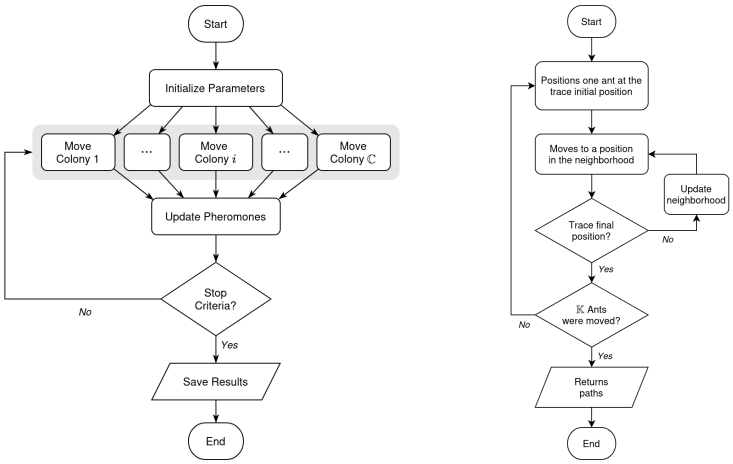
$$\Upsilon_k^{(c)} = \sum_{j \in \lambda_k^{(c)}} \sum_{u \in \mathbb{C}} f_c(u, j), \; u \neq c. \tag{7}$$

$$f_c(u, j) = \begin{cases} 1, & \text{if } j \in \lambda_{bs}^{(u)} \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

where $\lambda_{bs}^{(u)}$ is the best solution found so far by colony u. Finally, $\Gamma_k^{(c)}$ is the sum of the quadratic length differences between the current path and the best solutions found so far for the other colonies:

$$\Gamma_k^{(c)} = \sum_{u \in \mathbb{C}} (\Lambda_k^{(c)} - \Lambda_{bs}^{(u)})^2, u \neq c. \tag{9}$$

Figure 3 presents simplified flowcharts that illustrate the behaviour of the ACO PCB trace router. Figure 3a is the main loop of the algorithm, the movement of colonies are performed in parallel, each in its own process. The movement process for a colony is shown in Figure 3b. The update pheromone step is performed after all colonies have finished to move. The algorithm iterates until a stop criteria is reached. The stop criteria is the iteration number, but we also used an early stop strategy that finishes the execution when a result where all traces are length matched without crossings is found. This early stop strategy does not consider the minimal possible trace length, as this can only be known beforehand in cases that have been previously manually solved and do not involve a large number of traces. Therefore, results that meet the stop criteria are considered viable solutions for the problem within the scope of this work.



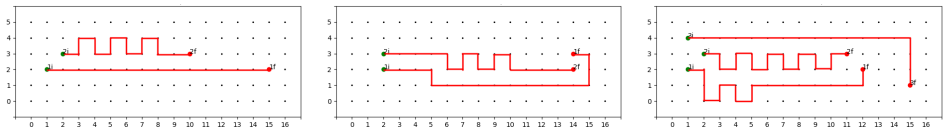
(a) Algorithm main loop. (b) Colony movement.

Figure 3. ACO PCB trace router flowcharts.

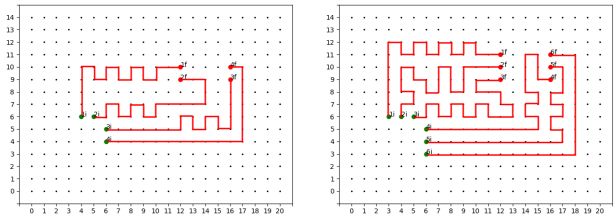
5 Experimental Results

The proposed algorithm was implemented in C++ and support scripts to generate graphs and analyze results were written in Python. The developed technique was tested and validated in five scenarios. The characteristics of optimal solutions are known for all scenarios. A result is considered optimal when there are no crossings, all traces have the same length and this length is the shortest possible length. Under this definition, multiple optimal solutions exist for each scenario. Figure 4 presents the five scenarios with an optimal solution found manually. In all scenarios, the algorithm was able to perform routing without crossovers and with length matching, results that meet at least these two criteria are considered viable solutions. The first scenario (Figure 4a) is the simplest. It consists of two traces, both with their initial and final positions on the same row. The second scenario (Figure 4b) is slightly more complex. This scenario consists of two traces with their starting and ending positions in the same columns but with the ending positions in different rows. One trace must navigate around the other while it uses the space between the starting and ending points to complete its routing. The third scenario (Figure 4c) combines the two previous scenarios. It has two traces aligned as in the first case and a third trace with misaligned initial and final positions; the trajectory of this trace needs to go around the other two. The fourth scenario (Figure 4d) has four traces and consists of a simplification of the routing between a Square Quad Flat Package (QFP) Integrated Circuit (IC) and a Dual In-Line Package (DIP) IC. The fifth and final scenario (Figure 4e) is the same as scenario four but with six traces.

Table 1 presents the average results of running the program one thousand times for each scenario. The first row indicates the percentage of optimal results found. The second row displays the percentage of results that meet at least the length matching and no crossing between traces objectives. These results fulfill the two most critical of the three objectives, thus they are deemed viable. The third row presents the average number of iterations the algorithm required to reach the stop criteria. The fourth row shows the average execution time in seconds that the algorithm took to reach the stop criteria, along with the standard deviation. The fifth and final row provides the average trace length of the results, also accompanied by the standard deviation. The closer the trace length is to



(a) Scenario 1. Optimal length = 14. (b) Scenario 2. Optimal length = 17. (c) Scenario 3. Optimal length = 17.



(d) Scenario 4. Optimal length = 18. (e) Scenario 5. Optimal length = 22.

Figure 4. All five test scenarios with optimal solutions found manually.

the minimal length defined for each scenario, the better. The execution stopped when a viable solution was found. The computer used had an Intel i9-12900KF CPU with 128 GB of memory.

Regarding hyperparameters, α and β were both set equal to two, giving equal influence to the pheromone trails and ant heuristic. The pheromone deposition constant Q was set to one. The evaporation rate ρ was set to 0.5, meaning that the pheromone is halved every iteration before it is updated. These values are typical for ACO algorithms. The number of ants per colony $\mathbb{K}$ was one hundred. This value was chosen to balance early exploration and execution times. In our tests, lower values increased the rate of suboptimal and non-viable solutions, meaning the algorithm more often got stuck in local optima, while higher values made execution times much longer. Nevertheless, there is some flexibility in this value. Finally, the weights ω_1, ω_2, and ω_3 that respectively balance the trace length minimization, crossings minimization, and length matching objectives were set at 10, 45, and 45. The length minimization objective was set to have a lower weight because in our tests, it tended to dominate the other objectives. The other two objectives were set with equal importance. Figure 5 shows five results obtained for each scenario. Results (a) to (q) are considered optimal, nevertheless all results presented are viable.

Table 1. Results obtained for each scenario.

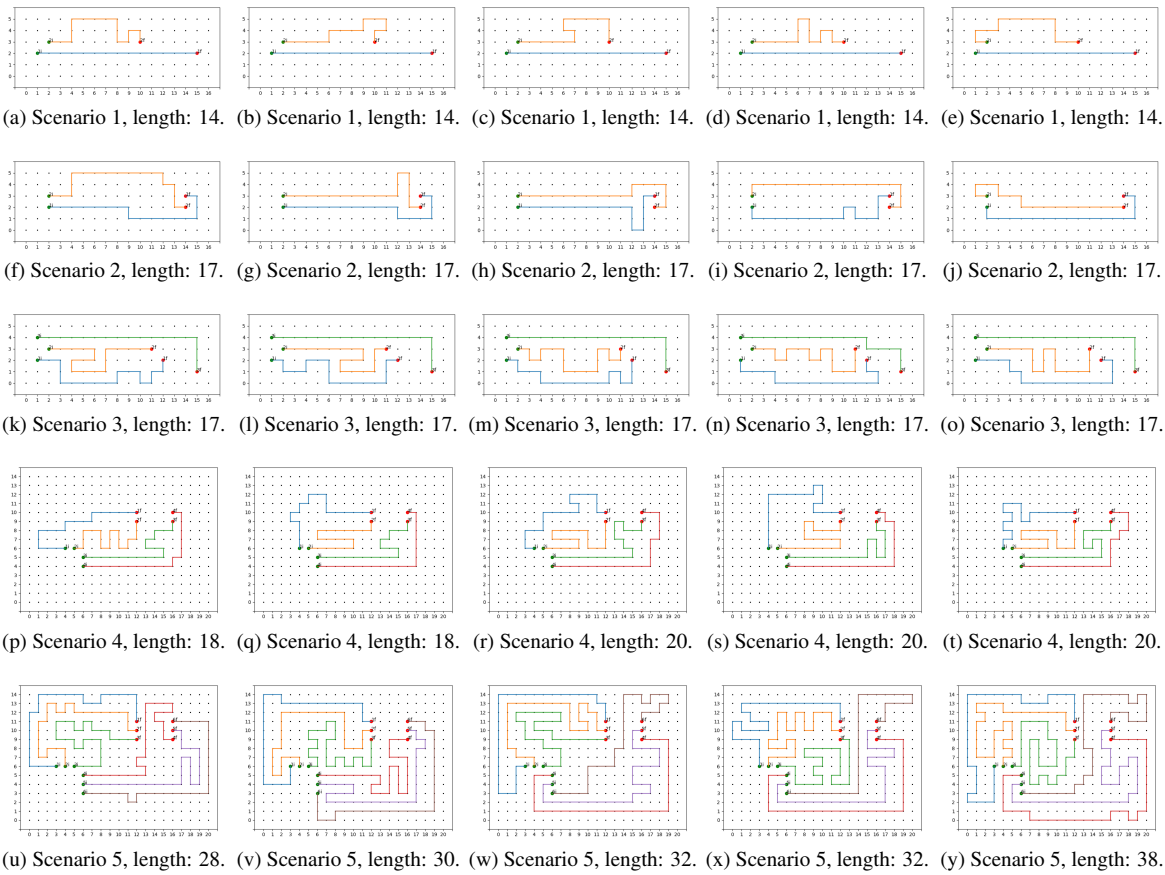
Stats	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Optimal (%)	64.9	26.8	23.6	1.3	0
No cross length matched (%)	100	82.1	75.4	98.8	6
Avg. Iterations	10	18542	38606	3995	96251
Avg. Execution time (s)	0.005 ± 0.012	24 ± 50	63 ± 67	22 ± 71	708 ± 135
Avg. Trace Length	14.9 ± 1.35	18.29 ± 2.9	18.88 ± 1.6	28.5 ± 6.1	37 ± 10

6 Conclusions

This paper introduced a variant of the ACO algorithm for PCB trace routing, aimed at minimizing crossings and ensuring length matching between traces. The algorithm performed best in Scenario 1, achieving 64.9% optimal solutions and 100% viable solutions with minimal iterations and short execution times. However, as complexity increased, performance declined. In Scenario 5, no optimal solutions were found, and only 6% of solutions were viable, requiring significantly more iterations and time. Scenarios 2 and 3 yielded around 25% optimal solutions and over 75% viable solutions. Despite the drop in performance in more complex cases, the algorithm proved effective in routing length-matched, crossing-free traces in most runs of the first four scenarios, demonstrating its potential for PCB routing.

However, the low number of optimal solutions and poor performance in Scenario 5, along with longer execution times in complex scenarios, highlight the need for further tuning of hyperparameters and algorithmic enhancements to make it suitable for real-world applications.

Future improvements include optimizing hyperparameters across all scenarios, incorporating additional heuristics to evaluate objectives, testing on more diverse scenarios, and allowing ants to make diagonal movements.



(a) Scenario 1, length: 14. (b) Scenario 1, length: 14. (c) Scenario 1, length: 14. (d) Scenario 1, length: 14. (e) Scenario 1, length: 14.

(f) Scenario 2, length: 17. (g) Scenario 2, length: 17. (h) Scenario 2, length: 17. (i) Scenario 2, length: 17. (j) Scenario 2, length: 17.

(k) Scenario 3, length: 17. (l) Scenario 3, length: 17. (m) Scenario 3, length: 17. (n) Scenario 3, length: 17. (o) Scenario 3, length: 17.

(p) Scenario 4, length: 18. (q) Scenario 4, length: 18. (r) Scenario 4, length: 20. (s) Scenario 4, length: 20. (t) Scenario 4, length: 20.

(u) Scenario 5, length: 28. (v) Scenario 5, length: 30. (w) Scenario 5, length: 32. (x) Scenario 5, length: 32. (y) Scenario 5, length: 38.

Figure 5. Five results obtained for each test scenario.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] H. Zhang, S. Krooswyk, and J. Ou. PCB design for signal integrity. In *High Speed Digital Design*, chapter 2, pp. 27–115. Morgan Kaufmann, 2015.

[2] V. Anand, V. Singh, and V. K. Ladwal. Study on pcb designing problems and their solutions. In *2019 International Conference on Power Electronics, Control and Automation (ICPECA)*, pp. 1–5, 2019.

[3] A. P. Alexander Weiler and A. Verma. *High-Speed Layout Guidelines*. Texas Instruments. Rev. A, 2017.

[4] S. Nath, S. Ghosh, and S. K. Sarkar. A novel approach to discrete particle swarm optimization for efficient routing in VLSI design. In *International Conference on Reliability, Infocom Technologies and Optimization*, 2015.

[5] J. A. Berlier. A parallel genetic algorithm for placement and routing on cloud computing platforms. Master's thesis, Virginia Commonwealth University, 2011.

[6] T. Badriyah, F. Setyorini, and N. Yuliawan. The implementation of genetic algorithm and routing Lee for PCB design optimization. In *International Conference on Informatics and Computing*, pp. 148–153. IEEE, 2016.

[7] K. Pandiaraj, P. Sivakumar, and R. Sridevi. Minimization of wirelength in 3D IC routing by using differential evolution algorithm. In *International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, pp. 1–5, Karur, India. IEEE, 2017.

[8] S. N. Bhargab Sinha and K. L. Baishnab. A hybrid RFD-ACO approach for routing optimization in VLSI physical design. In *Journal of Information and Optimization Sciences*, 2017.

[9] M. Doringo and T. Stützle. *Ant Colony Optimization*. The MIT Press, Cambridge, Massachussetts, 2004.