Evolutionary Algorithms for the Optimal Design of Robotic Cells: A Dual Approximation for Space and Time

Raúl-Alberto Sánchez-Sosa and Ernesto Chavero-Navarrete *

Posted Date: 10 July 2025

doi: 10.20944/preprints202507.0847.v1

Keywords: optimization; robotic cells; genetic algorithm; ant colony algorithm

 doi:10.20944/preprints202507.0847.v1

Article

Evolutionary Algorithms for the Optimal Design of Robotic Cells: A Dual Approximation for Space and Time

Raúl-Alberto Sánchez-Sosa [1] and Ernesto Chavero-Navarrete [2,*]

[1] Posgrado CIATEQ AC, Centro de Tecnología Avanzada

[2] CIATEQ AC, Centro de Tecnología Avanzada

* Correspondence: ernesto.chavero@ciateq.mx; Tel.: +52-442-196-15-00

Featured Application

Evolutionary algorithm-based computational tool for optimizing layout and trajectories in robotic cells, reducing cycle time and occupied area.

Abstract

The optimization of robotic cells is a key challenge in the manufacturing industry due to the need to maximize efficiency in limited spaces and minimize operation times. Traditional cell design methods often face challenges due to the high complexity and dynamic nature of real-world applications. In response, this study presents a dual approach to optimize both spatial design and traversal time in robotic cells, using bioinspired evolutionary algorithms. Initially, a genetic algorithm is employed to optimize the layout of the cell elements, reducing space usage and avoiding interferences between workstations. Subsequently, an ant colony optimization algorithm is used to optimize the robots' trajectories, minimizing cycle time. Through simulations and a digital model of the cell, key metrics such as total space reduction, operational time improvement, and productivity increase are evaluated. The results demonstrate that the combination of both approaches achieves significant improvements, enabling an average reduction of 21.19% in the occupied area and up to 20.15% in operational cycle time, consistently outperforming traditional methods. This approach has the potential to be applied in various industrial configurations, representing a relevant contribution in the integration of artificial intelligence techniques for the enhancement of robotic systems.

Keywords: optimization; robotic cells; genetic algorithm; ant colony algorithm

1. Introduction

In today's industrial landscape, where speed and efficiency are critical factors for success, optimizing robotic processes is not only a technical challenge but also a strategic necessity to maintain competitiveness. A robotic cell (RC), defined as an automated system composed of one or more robots, is designed to perform specific tasks in manufacturing and industrial production, such as assembly, welding and material handling. These configurations not only optimize workflow but also reduce human intervention, improve precision, and enhance efficiency in a controlled and safe environment [1,2].

The current trend in the design and improvement of RC focuses on optimizing operations to maximize productivity, reduce cycle times, and minimize operational costs [3]. This focus has led to the exploration of advanced methods aimed at enhancing overall performance and leveraging the capabilities of automated systems more effectively.

Traditionally, RC design has been carried out using methods that, while effective in some cases, often lack efficiency, flexibility, and scalability. Early designs required multidisciplinary teams to

 doi:10.20944/preprints202507.0847.v1

undergo iterative processes of review-adjustment, evaluating cell productivity through cost-benefit analyses [4]. Some authors proposed initiating the design process by determining the relative position of the robot and its assigned tasks, followed by incorporating machine dimensions, workstation locations, orientations, and specific operational characteristics of the implemented robot [5].

Other studies have addressed the optimization of cycle times and robotic trajectories using mathematical models based on kinematics and dynamics. These models calculate joint angles, velocities, and accelerations but are limited to well-defined and controlled systems [6–8].

Classical optimization algorithms, such as gradient descent, have been employed to adjust trajectories and minimize energy or time, while linear programming has been used to optimize dynamic work areas. However, these approaches have shown limitations in scalability and efficiency, particularly in nonlinear or large-scale problems [9]. On the other hand, simulations have served as practical tools to optimize trajectories and spatial layouts before the physical implementation of robots. These simulations allow for testing multiple configurations without disrupting production, identifying potential collisions, and significantly reducing testing time and costs. Nevertheless, their effectiveness heavily relies on the fidelity of the simulated model [10,11].

Additionally, expert systems, generally based on rule-based models, have enabled the integration of historical engineering data to generate process design alternatives that consider both performance requirements and the specific constraints and specifications of robots and components [12,13].

Feedback-based optimization operates in real-time using sensors, such as cameras and force or position sensors, to dynamically adjust robotic trajectories. Algorithms analyze sensor data to correct paths and avoid collisions. While this approach adapts well to dynamic environments, its response is slower due to higher computational demands compared to preprocessed solutions [14,15].

Meanwhile, the combination of virtual reality and digital twins is revolutionizing the design, analysis, and optimization of RC. Through 3D virtual models that accurately replicate all physical components in a digital environment, these tools allow for the simulation of various spatial configurations before making physical changes, providing an immersive real-time experience. These models are particularly useful for identifying ergonomic or accessibility issues that traditional algorithms might overlook. However, despite their ability to test multiple configurations, the number of scenarios that can be simulated is limited by computational resources and system capabilities, preventing guarantees of global optimization [16–18].

Recent research has explored advanced algorithms, such as mixed-integer programming and evolutionary algorithms, to address complex optimization problems. Genetic algorithms (GA) have been successfully applied to solve production flow problems with blocking constraints and variable processing times [19], as well as hierarchical planning focused on optimizing postures and movements [20].

In [21], a heuristic algorithm optimizes the placement of workstations, modeling the robot through homogeneous transformation matrices, representing components with cylindrical envelopes, and identifying collisions via 2D circular projections. Trajectories are segmentally adjusted after relocating interference points. Furthermore, approaches based on GA, particle swarm optimization (PSO), ant colonies optimization (ACO), and differential evolution have proven effective in optimizing criteria such as workspace layout, operational time, and system manipulability, with PSO algorithms standing out for their high-quality solutions [22].

Evolutionary algorithms (EA), however, present certain limitations, including high computational costs, reliance on well-defined objective functions, and challenges in handling constraints. Additionally, they may produce solutions that are difficult to interpret, restricting their effectiveness and applicability in complex or resource-constrained environments. For instance, an algorithm might suggest placing two workstations at opposite corners of the cell to minimize interferences, but this layout could be impractical for human supervision or maintenance access. The quality of results is closely tied to the proper selection of parameters, such as population size, mutation rates, and crossover rates. Nevertheless, they excel at exploring extensive solution spaces,

 doi:10.20944/preprints202507.0847.v1

even in complex and nonlinear problems with multiple variables, demonstrating effectiveness in scenarios with uncertain or dynamically changing parameters [23].

While simulations enable visualization of system behavior, facilitating the identification of issues such as collisions, inefficient trajectories, or bottlenecks, EA stands out for their capacity for global exploration and optimization. Several authors suggest that combining both tools may lead to better results [21,22].

This article proposes an integrated approach for optimizing the design of workspaces and cycle time in RC, addressing the limitations of current methods and tools. The approach involves using GA to optimize workspace distribution by determining the optimal placement of workstations, followed by an ACO algorithm to identify the shortest process trajectory. Together, these algorithms form an accessible design tool that does not require advanced knowledge for implementation, reducing reliance on highly specialized experts.

The method is implemented and evaluated in practical industrial scenarios, with results validated using Mitsubishi's RT TOOLBOX3 PRO simulation software [24]. Additionally, innovative criteria are introduced for evaluating RC designs, emphasizing flexibility, resilience, and reconfigurability. This approach integrates emerging technologies with advanced algorithms to deliver scalable, efficient, and practical solutions applicable to both academic research and industrial operations.

2. Materials and Methods

2.1. Robotic Cell

A RC is an integrated system in manufacturing and industrial automation that employs one or more robots, along with equipment and tools, to perform specific tasks such as assembly, welding, painting, material handling, and inspection. These cells are designed to enhance efficiency, precision, and safety in production processes. They are particularly useful for reducing labor costs, improving product quality, and operating in hazardous environments. Additional benefits of a RC include reducing human errors, enabling continuous operation, and improving product consistency. However, their implementation involves a significant investment, making it crucial to optimize their programming, particularly in transportation and processing operations, to avoid bottlenecks and maximize performance [25].

The layout of a RC is the structural design that organizes robots, equipment, tools, and workstations within a cell to optimize workflow, minimize downtime, and maximize efficiency. Achieving an efficient design requires consideration of factors such as material flow, optimal space utilization, safety, flexibility to adapt to changes, and equipment integration. Layouts are classified into configurations based on process requirements and space constraints. Among the most common are:

- Linear layout: Equipment and robots are arranged sequentially for high-volume processes.
- U-shaped or open layout: Optimizes space, facilitates quick access between stations, and offers greater flexibility for customized configurations.
- Cellular or circular layout: Robots operate from a central position toward distributed stations, suitable for continuous or loop-based cyclic processes [26].

Proper planning of these aspects contributes to improving costs, productivity, and the overall quality of the automated system. Figure 1 illustrates the typical configurations of an RC.

Work area optimization focuses on strategically organizing the physical space to ensure a continuous workflow. Meanwhile, cycle time optimization reduces the time required to complete tasks by eliminating unnecessary movements. Together, these strategies enhance production speed, lower operational costs, and ensure effective and safe processes [23].

 doi:10.20944/preprints202507.0847.v1

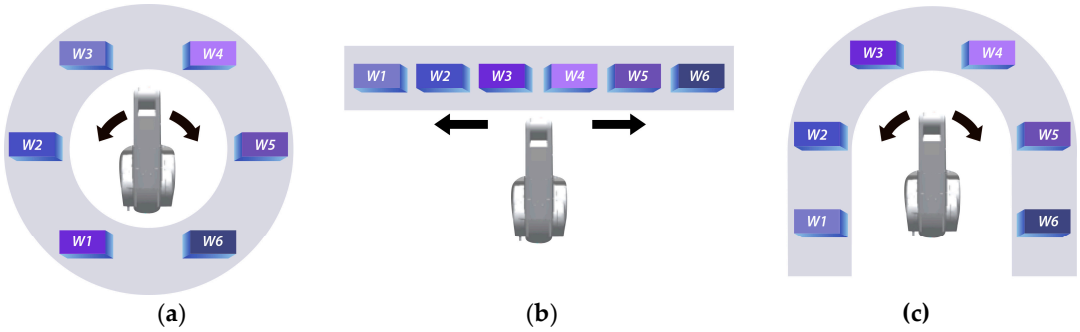


Figure 1. Configurations of a RC: (a) Circular: facilitates quick access between stations and optimizes space usage in compact areas. (b) Linear: enables a logical and sequential flow of materials or products, ideal for assembly line processes. (c) U-shaped: allows the workflow to return to the starting point, reducing downtime and minimizing material movement.

2.2. Genetic Algorithm

A GA is an evolutionary algorithm inspired by natural selection and the biological processes of reproducing the fittest individuals. Initially, the GA generates a population of random candidate solutions, referred to as individuals or chromosomes. Each chromosome consists of a set of genes that represent the problem's variables. The evolutionary process is governed by an iterative mechanism in which a fitness function evaluates how well a solution addresses the problem. The higher the fitness, the greater the probability of the solution being selected for the next generation [27].

GAs employ three main operators to drive the evolutionary process:

Selection: This process determines which chromosomes will contribute to the next generation. The most common methods include Roulette Wheel, where the probability of selection is proportional to an individual's fitness; Tournament, in which a random group of individuals is selected, and the fittest advances; and Rank Selection, which orders individuals by their fitness and assigns selection probabilities proportional to their rank.

Crossover: This operator combines pairs of chromosomes to produce new offspring. Crossovers can occur at one or two random points or follow a fixed probability to determine the crossover location.

Mutation: This introduces variability by randomly altering one or more genes in a chromosome. Mutation prevents the population from becoming trapped in local optima and enhances exploration of the search space [28].

Figure 2 illustrates the cycle of a GA, represented as a circular flowchart. It begins with an initial population of chromosomes and progresses through key stages such as fitness evaluation, selection of the fittest individuals, crossover to generate new chromosomes, and mutation to introduce variability. These stages are repeated iteratively until a stopping criterion is met, such as a predefined number of generations or no change in the fitness value, highlighting the continuous evolutionary process toward optimal solutions.

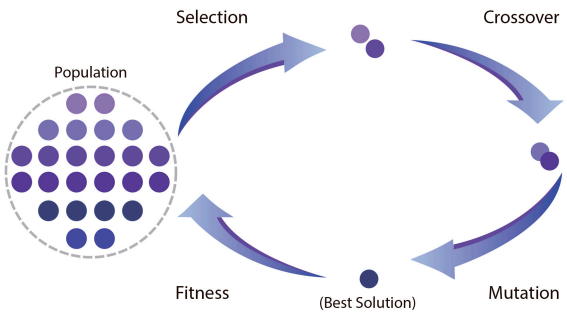


Figure 2. The GA cycle illustrates the core stages of the evolutionary process: initialization of the population, evaluation of individuals, selection of the best candidates, application of genetic operators (crossover and

 doi:10.20944/preprints202507.0847.v1

mutation), and generation of a new population. This cycle is repeated iteratively until a predefined stopping criterion is satisfied.

2.3. Ant Colonies Algorithm

ACO is a search technique inspired by the collective behavior of ants in finding short paths to a food source. Ants initially move randomly until they discover food, then return to their colony while depositing pheromones along their path. These pheromones serve as a guide for other ants to follow the same route, allowing them to find the food source without moving randomly. If multiple paths to the food source exist, shorter paths will accumulate higher pheromone concentrations, making them more attractive. Conversely, longer paths will gradually lose pheromones due to their volatility. In this way, the ants collectively determine the shortest route [29].

In ACO, artificial ants search for solutions within a defined search space. Pheromones are represented as numerical values that reinforce the best solutions, while suboptimal ones are discarded through an evaporation process. With successive iterations, the algorithm converges towards the optimal or near-optimal solution, emulating the cooperative and adaptive behavior of a colony of ants. Mathematically, the problem is represented as a graph where nodes correspond to states and edges represent transitions between them. An initial uniform amount of pheromone is assigned to all edges. A set of artificial ants constructs solutions by traversing the graph, starting from an initial node and building a complete solution. The probability of selecting an edge depends on the higher pheromone levels and problem-specific information, such as distance or cost [29].

The probability P_{ij} of an ant selecting edge (i,j) is calculated as [30]:

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha}[\mu_{ij}]^{\beta}}{\sum_{k \in candidates}[\tau_{ij}]^{\alpha}[\mu_{ij}]^{\beta}} \quad (1)$$

where:

τ_{ij} : Pheromone level on edge (i,j).

μ_{ij} : Heuristic value associated with edge (i,j).

α : Importance of pheromones.

β : Importance of heuristics.

This probabilistic selection ensures a balance between exploration of new paths and exploitation of existing high-quality solutions. Once the ants complete their solutions, the quality of each solution is evaluated based on the problem's objective function. The pheromone levels on the transitions used in the best solutions are updated by adding pheromones. Meanwhile, the pheromone levels on other transitions are reduced by an evaporation factor, preventing stagnation in suboptimal solutions. The pheromone update is mathematically expressed as [30]:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij} \quad (2)$$

where:

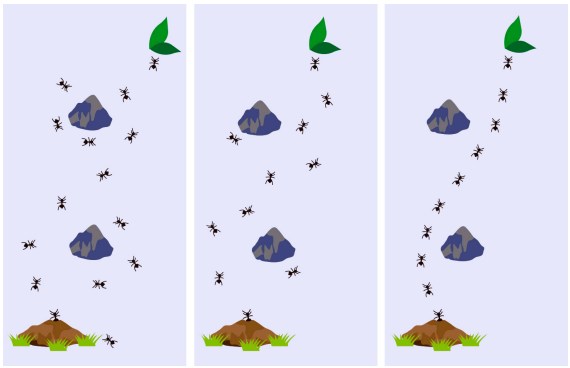
$\Delta\tau_{ij}$: Increment in pheromone on edge (i,j), typically proportional to the inverse of the solution cost or another performance metric.

ϱ : Evaporation rate, controlling the reduction of pheromone levels.

The above steps are repeated for a fixed number of iterations or until a stopping criterion is met, such as reaching an optimal solution. Figure 3 illustrates the ACO cycle through a circular flowchart, highlighting the main stages of the process.

(a) (b) (c)

 doi:10.20944/preprints202507.0847.v1



(a) (b) (c)

Figure 3. Key stages of the ACO: (a) Initialization of the environment, (b) solution construction by the ants, solution evaluation, and pheromone update, and (c) repetition of the cycle until a stopping criterion is met.

2.4. Methodology

This study proposes a dual-approach evolutionary methodology for the optimal design of RC, aimed at simultaneously improving spatial distribution (layout) and temporal efficiency (robot trajectory). The solution is structured in two main stages: the first phase employs a GA to optimize the placement of workstations within the cell, while the second phase uses an ACO algorithm to determine the most efficient trajectory of the robotic arm between those stations.

2.4.1. Geometric Modeling of the Robotic Cell Environment

The physical environment of each RC consists of a specific set of workstations with variable dimensions, an industrial robot, and a defined operating area. Each of these elements is modeled as a rectangular polygon, defined as:

$$workstation_i = \{(x_{min}, y_{min}), (x_{max}, y_{max})\} \quad (3)$$

$$RobotBase = \{(x_1, y_1), (x_2, y_2)\} \quad (4)$$

where (x_{min}, y_{min}) and (x_{max}, y_{max}) correspond to the opposite corners of the diagonal that delimit the area occupied by element i, as illustrated in Figure 4. This representation facilitates the calculation of areas, collision detection, and verification of spatial constraints.

To ensure the operational feasibility of the system, geometric constraints are incorporated, including exclusion zones, pedestrian pathways, physical interferences, and safety margins between the various components of the cell.

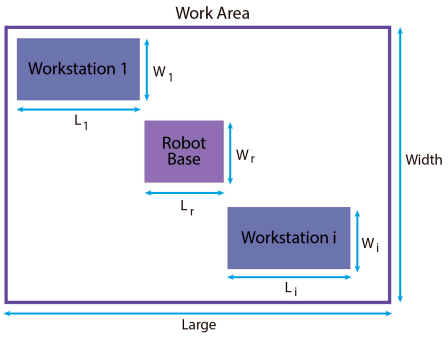


Figure 4. Geometric modeling of workstations, robot base, and defined operating area for the spatial configuration of the RC.

 doi:10.20944/preprints202507.0847.v1

2.4.2. Design Vector and Objective Function Formulation

Once the physical environment of the RC has been defined, the problem is formulated as a combinatorial multi-objective optimization, aiming to determine the optimal spatial configuration of the workstations. The goal is to simultaneously minimize the total occupied area and the physical interferences between components.

Each workstation is encoded as a gene within a continuous design vector. Since this is a real-valued representation, each gene takes coordinates (x_i, yi) within the allowed domain of the workspace:

$$Gen_i = (x_1, y_1) \in [0, L]\text{x}[0, W] \tag{5}$$

where L and W are the dimensions of the available workspace. A complete individual (i.e., a candidate solution) is defined as:

$$Individual = \{Gen_1, Gen_2, \dots, Gen_n\} \tag{6}$$

This modeling generates an initial population randomly distributed according to a uniform distribution. Based on this encoding, the algorithm evaluates two objective functions:

- Minimization of the bounding rectangle area that contains all workstations:

$$f_{area} = (X_{max} - X_{min}) \cdot (X_{max} - X_{min}) \tag{7}$$

- Minimization of the number of interferences (geometric overlaps) between workstations:

$$f_{interference} = \sum_{i,j} \left(x_{i\,max} \geq x_{j\,min} \wedge x_{j\,min} \geq x_{i\,min} \wedge y_{i\,max} \geq y_{j\,min} \wedge y_{j\,min} \geq y_{i\,min}\right) \tag{8}$$

These objectives are combined in a weighted composite function:

$$f_{comb} = w_1 \cdot f_{area} + w_2 \cdot f_{interference} \tag{9}$$

where weights w_1 and w_2 allow adjusting the relative importance of each objective, prioritizing in this case the elimination of physical interferences.

2.4.3. Evolutionary Operators

Once the population is evaluated using the composite fitness function, individuals are selected to participate in the crossover stage. A binary tournament selection strategy is employed due to its balance between computational simplicity and effective selective pressure.

This process involves performing $\lambda=N/2$ independent tournaments, where N is the population size. In each tournament, two individuals are randomly selected using uniform sampling, and their performance is compared based on the objective function, which weighs both the occupied area of the layout and the number of interferences between stations. The better-performing individual is chosen as a parent.

The proposed GA relies solely on the classic one-point crossover operator. This choice is based on two key considerations:

1. Structured exploratory capability: crossover enables the efficient recombination of promising solutions, facilitating the transfer of useful substructures (gene blocks) between individuals without significantly disrupting the integrity of the layout.
2. Population stability: by omitting random mutation operators, uncontrolled dispersion in the search space is avoided, allowing smoother convergence toward feasible regions of the design space.

The crossover operator is applied with a probability Pc, generating two offspring from a randomly selected point n within the gene vector. The first offspring inherits the first n genes from the first parent and the remaining genes from the second parent; the second offspring is constructed in reverse. This scheme maintains structural diversity without the need for additional operators,

 doi:10.20944/preprints202507.0847.v1

which is particularly beneficial in problems where the station topology directly affects the feasibility of the resulting design.

2.4.4. General Framework of the Proposed Genetic Algorithm

The GA was implemented using the Python programming language [31], version 3.5, within the integrated development environment (IDE) Visual Studio Code [32], version 1.91. Experimental tests were conducted on a computer equipped with an 11th Gen Intel® Core™ i7-11800H processor running at 2.30 GHz and 64 GB of RAM.

The overall workflow of the algorithm is summarized in Algorithm 1, which outlines the general logic of the evolutionary process from the random generation of the initial population to the identification of the best solution found after applying the defined evolutionary operators.

Algorithm 1: GA for RC layout optimization

```
# Inputs:
# N: population size
# G: number of generations
# Pc: crossover probability
# w1, w2: weights for area and interference
# domain: coordinate bounds for station placement

def genetic_algorithm(N, G, Pc, w1, w2, domain):
    population = initialize_population(N, domain)     # Initialize population
    for individual in population:     # Evaluate initial population
        area = evaluate_area(individual)
        interference = evaluate_interference(individual)
        individual.fitness = w1 * area + w2 * interference
    for generation in range(G):     # Evolutionary loop
        new_population = []
        while len(new_population) < N:     # Tournament selection
            parent1 = tournament_selection(population)
            parent2 = tournament_selection(population)
            if random.random() < Pc:     # One-point crossover
                child1, child2 = one_point_crossover(parent1, parent2)
            else:
                child1, child2 = parent1.copy(), parent2.copy()
            new_population.extend([child1, child2])
        for individual in new_population:     # Evaluate new population
            area = evaluate_area(individual)
            interference = evaluate_interference(individual)
            individual.fitness = w1 * area + w2 * interference
        population = new_population     # Replace old population
    best = min(population, key=lambda ind: ind.fitness)     # Return best solution
    return best
```

 doi:10.20944/preprints202507.0847.v1

2.4.5. Sensitivity Analysis and Parameter Tuning of GA

To validate the robustness of the GA and select efficient parameter configurations, a sensitivity analysis was performed on key evolutionary parameters: population size, number of generations, crossover probability (P_c), and the weights w_1 and w_2 used in the composite fitness function. The aim was to assess how these values affect layout quality, station interference, and execution time.

This analysis consisted of multiple algorithms runs, where each parameter was varied systematically while holding the others constant. The impact on performance metric, such as convergence time, number of collisions, and total area were recorded for each setting.

Crossover Probability (P_c): This parameter controls how frequently the recombination operator is applied to the selected parents. Since crossover is the only evolutionary operator used, a high P_c (between 0.9 and 1.0) is recommended to promote broad exploration and prevent premature convergence. Slightly lower values (e.g., 0.8 or 0.85) may be considered if population diversity declines.

Population Size and Number of Generations: These parameters were empirically tuned based on algorithm performance under different test cases. Larger populations improve genetic diversity but increase computation time. Similarly, more generations enable deeper exploration but at higher computational cost. Tests showed that using between 30 and 100 individuals and 50 to 150 generations provided a good balance between solution quality and processing efficiency.

Weights w_1 and w_2 in the Composite Function: Each candidate solution was evaluated using a weighted fitness function (9) that combines two objectives: minimizing the total layout area (f_{area}) and minimizing physical interferences between stations ($f_{interference}$). The following weight combinations were tested: $w_1 = 1$, $w_2 = 100$; $w_1 = 1$, $w_2 = 500$; $w_1 = 1$, $w_2 = 1000$.

The final selection was based on solutions that avoided interference while maintaining compact layouts. Interference reduction was prioritized due to its critical impact on the physical feasibility of the RC.

Table 1 presents a representative subset of the results obtained, specifically the top 10 scenarios evaluated during the sensitivity analysis. Each row shows the parameter combinations that delivered the best performance in terms of layout compactness, interference elimination, and execution time. This selection highlights the most efficient configurations and provides empirical support for the final parameter choices used in the GA implementation.

Table 1. Best parameter combinations of GA obtained from the sensitivity analysis.

Population	Generations	P_c	w_1	w_2	area (u^2)[1]	Interferences	Execution time (min)
60	100	0.95	1	500	81	0	15.12
60	100	0.95	1	500	95	0	17.25
60	150	0.95	1	1000	71	1	24.92
30	50	0.90	1	100	85	1	23.04
100	100	1.00	1	500	78	2	24.92
30	150	0.90	1	1000	90	2	27.16
60	50	0.95	1	100	77	3	15.69
100	50	1.00	1	100	92	3	24.92
30	100	0.90	1	500	95	3	25.20
100	150	1.00	1	1000	94	3	31.29

[1] u^2 denotes square units relative to the coordinate system of the work area.

The sensitivity analysis enabled the identification of efficient configurations of the GA in terms of both solution quality and execution time. The most effective parameter combination was achieved with a population size of 60, 100 generations, a crossover probability P_c=0.95, and weights w_1=1 and w_2=500. This configuration produced a layout with an occupied area of 81 square units, no interferences between stations, and an execution time of only 15.12 minutes.

This result shows that moderate parameterization can be sufficient to reach optimal solutions (in terms of population size and number of generations), provided that adequate selective pressure

 doi:10.20944/preprints202507.0847.v1

is maintained through crossover and that geometric feasibility is prioritized via proper weighting in the objective function.

2.4.6. Graph Model

At this stage, the goal is to optimize the robotic arm's path between the workstations, based on the spatial configuration previously determined by the GA. To achieve this, a fully connected directed graph is constructed, where each node represents a workstation located at coordinates (x, y), and each edge corresponds to a possible path between two stations.

The graph is formally defined as:

$$G = (V, E) \tag{10}$$

where:

$V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes representing the workstations in the optimal layout produced by the GA.

$E = \{(v_i, v_j) \mid i \neq j\}$ is the set of edges representing all possible transitions between station pairs.

Each edge (v_i, v_j) is associated with a heuristic cost η_{ij}, which can be computed based on the Euclidean distance between stations or estimated travel time considering the robot arm's constraints, such as maximum speed and acceleration.

The graph is implemented using an adjacency matrix, where each cell stores the heuristic value and the corresponding pheromone level for that edge. This graph model serves as the foundation for applying the ACO algorithm, which aims to find the most efficient route visiting each station exactly once while minimizing the robot's total cycle time. Figure 5 shows an example of a graph with four stations, where the heuristic cost and pheromone level associated with each edge are illustrated.

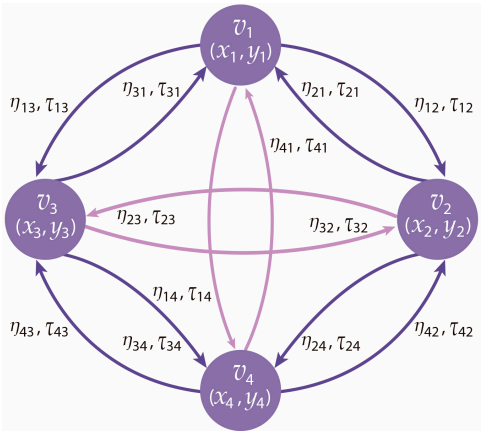


Figure 5. Directed graph representation with workstations, heuristic costs, and pheromone levels for each trajectory.

2.4.7. Definition of ACO Parameters

The performance of ACO largely depends on the appropriate configuration of its key parameters, which govern the balance between exploration of the solution space and convergence toward optimal solutions. The following are the main components that determine the system's behavior:

Number of ants (m): Each ant represents a tentative solution that visits all stations exactly once. One ant per station was used ($m = n$), allowing each ant to start from a different station to enhance diversity in graph exploration.

Initial pheromone (τ_0): A uniform pheromone value was assigned to all edges at the start of the algorithm, typically ranging from 1 to 2. This neutral setting enables the evolutionary process to guide path selection based on accumulated performance.

 doi:10.20944/preprints202507.0847.v1

Evaporation rate (ρ): Controls how quickly pheromone values decrease in each iteration. High values of ρ (0.4–0.5) promote exploration, while low values (0.1–0.2) promote exploitation and faster convergence. In this work, values between 0.1 and 0.5 were evaluated.

Heuristic (η_{ij}): Each edge (i, j) has a heuristic value based on the estimated travel time, considering robotic arm constraints such as maximum speed and acceleration. It is calculated as:

$$\eta_{ij} = \frac{1}{time_{ij}} \tag{11}$$

Higher values indicate more favorable paths.

Relative importance of pheromone and heuristic (α, β): These parameters control how much the ant's decisions are influenced by accumulated pheromone α versus the local heuristic β. In this implementation, values between 1 and 2 were used for both, promoting a balanced influence between past success and travel efficiency.

Number of iterations: A maximum of 1000 cycles was set as the stopping criterion. Alternatively, the algorithm halts if no substantial improvement is observed over 20 consecutive iterations.

A dual strategy combining intensification and diversification was implemented to guide the search process. Intensification reinforces the most promising paths by increasing pheromone levels on successful routes, while diversification promotes exploration through pheromone evaporation and the deployment of multiple ants with randomized initial routes. This balance is essential to prevent premature convergence to suboptimal solutions and to avoid excessive exploration without meaningful improvement.

2.4.8. General Scheme of the Proposed ACO

Based on the previous sections, the pseudocode summarizing the general logic of the ACO is presented. This includes the processes of route construction, total travel time evaluation, pheromone level updating, and identification of the best solution found.

The implementation was carried out using the same hardware and software resources previously employed in the execution of the GA. The complete procedure is described in Algorithm 2.

Algorithm 2: ACO for minimizing robot path time between stations

```
# Inputs:
# m: set number of ants
# E: iterations
# t0: initialize pheromone
# p: evaporation rate
# a: pheromone influence
# b: heuristic influence
# Q: pheromone deposit factor

initialize_pheromone_matrix(t0)     # Set initial pheromone value τ0
calculate_heuristic_matrix(n)     # Compute heuristic values based on travel time

for interation in range(E):     # ACO main loop
        solutions = []     # Store all constructed paths and their travel times
        for ant in range(m):     # Each ant constructs a solution
                current_node = randomly_select_start_station()
                visited_nodes = [current_node]
```

 doi:10.20944/preprints202507.0847.v1


```
            tour = [current_node]
            total_time = 0
            while not all_nodes_visited(visited_nodes):     # Construct the full tour
                next_node = select_next_node(
                    current_node,
                    visited_nodes,
                    pheromone_matrix,
                    heuristic_matrix,
                    a,b
                )
                total_time += estimate_time(current_node, next_node)
                tour.append(next_node)
                visited_nodes.append(next_node)
                current_node = next_node
            solutions.append((tour, total_time))     # Save the solution
        pheromone_matrix *= (1 - p)     # Pheromone evaporation
        best_solution = select_best_solution(solutions)     # Pheromone update
        for i in range(len(best_solution[0]) - 1):
            from_node = best_solution[0][i]
            to_node = best_solution[0][i+1]
            pheromone_matrix[from_node][to_node] += Q / best_solution[1]
        if convergence_criteria_met():        # Check for convergence or stop criteria
            break
        return best _solution
```

2.4.9. Sensitivity Analysis and Parameter Tuning of ACO

To assess the influence of key parameters in ACO on path optimization performance, a systematic sensitivity analysis was carried out. The parameters considered included: the selection probability associated with pheromone levels (α), ranging from 1.0 to 2.0; the heuristic influence based on estimated distance (β), also ranging from 1.0 to 2.0; the pheromone evaporation rate (ϱ), evaluated between 0.1 and 0.5; the initial pheromone quantity (Q), with values between 1.0 and 2.0; the number of ants (m), set equal to the number of workstations (m = N); and the maximum number of iterations (T), limited to 100 cycles.

Representative combinations of these parameters were defined, while maintaining the geometric conditions of the RC fixed, based on the layout previously optimized by GA. Each configuration was evaluated based on the total time required for the robot to visit all stations, considering kinematic constraints such as maximum speed and acceleration.

The results revealed configurations that support convergence toward more efficient paths. Specifically, higher values of α (between 1.5 and 2.0) and moderate evaporation rates (ϱ between 0.2 and 0.4) led to more robust solutions by balancing the exploitation of promising paths and the exploration of new alternatives. Additionally, it was confirmed that assigning one ant per station yields good results without significantly increasing computation time. A summary of the evaluated scenarios and the best results obtained is presented in Table 2.

 doi:10.20944/preprints202507.0847.v1

Table 2. Best parameter combinations of ACO obtained from the sensitivity analysis.

Pheromone levels (α)	heuristic influence (β)	pheromone evaporation (p)	Initial pheromone τ_0	ants (m)	Actual Interaction	Best time (seg)
1.5	2.0	0.3	1.5	6	75	11.28
1.5	2.0	0.2	1.5	6	79	12.16
2.0	1.5	0.3	2.0	6	83	12.54
1.5	2.0	0.3	1.0	6	87	12.77
2.0	1.5	0.3	1.0	6	92	12.86
1.5	2.0	0.3	2.0	6	98	12.96
2.0	1.5	0.1	1.0	6	100	13.52
1.0	1.5	0.1	1.5	6	100	13.57
1.5	2.0	0.5	1.0	6	100	13.62
1.5	1.0	0.2	2.0	6	100	13.66

The results presented in the ACO Sensitivity Table clearly demonstrate how different parameter combinations significantly affect the algorithm's performance, particularly in terms of the robot's total travel time. The most efficient configurations were observed for values of $\alpha = 1.5$, $\beta = 2.0$, $\varrho = 0.3$, and an initial pheromone level of $\tau_0 \approx 1.5$. With this parameter setting, the algorithm was able to minimize the total path time to below 11.28 seconds, representing highly favorable performance in terms of temporal efficiency.

3. Results

The proposed methodological approach is validated through its application to three RC currently operating in an industrial environment, corresponding to assembly, machining, and screwing processes. Each cell is treated as an independent case study, allowing for the assessment of the robustness, flexibility, and generalization capability of the applied evolutionary algorithms.

For each case, a comparison is made between the original configuration of the RC and the optimized layout generated by the proposed algorithm, considering two main performance metrics:

- Occupied area of the RC, measured as the minimum rectangular area that encloses all the stations.
- Total cycle time, which refers to the time required to complete a full sequence of operations between stations.

Additionally, the results are compared against the estimations provided by the commercial software Mitsubishi RT TOOLBOX3 PRO, which is widely used in the industry for robot trajectory simulation and evaluation. This comparative approach demonstrates the effectiveness of the proposed model not only in relation to current operating conditions but also when benchmarked against specialized and widely accepted industrial tools.

3.1. Case Study 1: Optimization of a Robotic Machining Cell

The CNC machining center is designed to manufacture three different types of shafts, each with variations in length and diameter. The RC is composed of one input station, where raw parts are loaded, and three output stations, where the finished products are placed and classified according to the type of shaft produced. The CNC machine includes a circular exclusion zone that the robot cannot reach due to its proximity to the base, which is modeled as a square-shaped area. Table 3 presents the dimensions of each workstation, which will be used as a reference for applying the proposed optimization methodology.

Table 3. Dimensions of machining stations.

Stations	length (mm)	Width (mm)
CNC	750	600

 doi:10.20944/preprints202507.0847.v1

Station 1	350	325
Station 2	400	150
Station 3	275	225
Sation 4	260	197
Robot base	300	300

The RC is equipped with a six-axis industrial robot (RV8CRLD-S15 M, Mitsubishi Electric, Tokyo, Japan), specifically designed for handling tasks in confined spaces. This unit supports a payload of up to 8 kg, provides a maximum reach of 931 mm, and features a repeatability of ±0.02 mm, making it suitable for precision operations in compact environments.

Figure 6 shows the robot installed within the RC, along with the technical specifications required as input for the proposed algorithm.

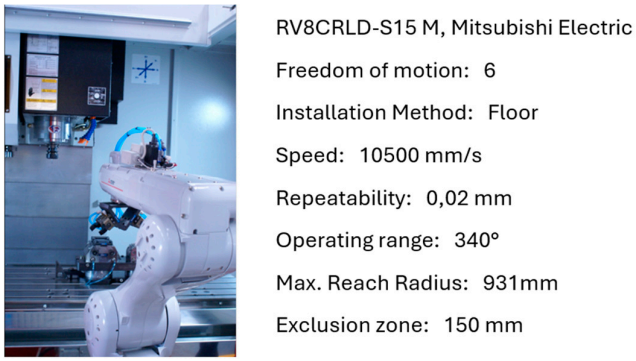


Figure 6. RV-8CRL-D industrial robot installed in the RC with technical specifications used in the optimization model.

For this case study, the maximum workspace considered by the algorithm corresponds to the current layout of the RC, since the objective is to optimize this arrangement by reducing the utilized area. The operational environment measures 847.5 mm in width and 1737 mm in length, representing a total area of 1.47 m². Based on the input parameters, which include the current spatial arrangement of the workstations and the kinematic capabilities of the robot, the proposed optimization methodology was executed. Figure 7 shows the current configuration of the RC prior to the application of the optimization.

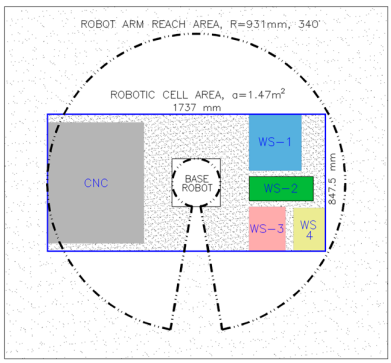


Figure 7. Current configuration of the machining RC, showing the spatial layout of workstations and the maximum workspace considered for optimization.

The GA evaluated multiple possible configurations within the defined workspace, considering both geometric constraints and efficiency criteria related to the robot's movement. As a result, a new, more compact workstation layout was obtained, allowing the robot to access all stations without excessive trajectories or collisions. The final configuration occupies an area of 750 mm by 1634 mm (equivalent to 1.25 m²), representing a 14.97% reduction compared to the original area. Figure 8 shows this optimized layout generated by the GA.

 doi:10.20944/preprints202507.0847.v1

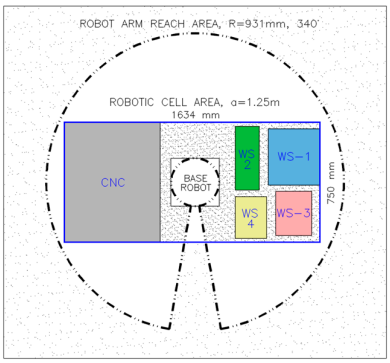


Figure 8. Optimized layout of the RC after applying for the GA.

After determining the optimal spatial configuration of the RC using the GA, the ACO was applied to compute the robot's shortest trajectory between stations. To quantify its performance, a comparative assessment was conducted across three reference scenarios: (1) the actual cycle time recorded during the current operation of the cell, (2) the cycle time estimated by the Mitsubishi RT TOOLBOX3 PRO simulation software using the new layout, and (3) the optimized cycle time resulting from the execution of the ACO algorithm on the same configuration.

Table 4 summarizes the results obtained in each case, providing a quantitative evaluation of the improvements achieved by the ACO in terms of cycle time reduction. This comparison serves to validate the efficiency of the proposed model with respect to conventional simulation-based approaches.

Table 4. Cycle time comparison between the current configuration, optimization AG and simulation software, and the ACO result.

Trajectory	Current Transfer Time (s)	AG+MRT Transfer Time (s)	AG+ACO Transfer Time (s)
1 to CNC	4.56	4.46	4.12
2 to CNC	3.45	2.57	2.37
3 to CNC	3.89	4.03	3.71
4 to CNC	3.71	2.78	2.57
Total Time	15.61	13.84	12.77

The results presented in Table 4 reveal a substantial improvement in the performance of the RC following the application of the proposed optimization methodology. Compared to the current cell configuration, the total transfer time was reduced from 15.61 seconds to 13.84 seconds by implementing the new layout optimized by the GA and simulating the transfer time using the Mitsubishi RT TOOLBOX3 PRO software, representing an 11.34% improvement. This gain was further enhanced by applying the ACO to the same spatial configuration, which reduced the total time to 12.77 seconds. Overall, this outcome represents an 18.19% reduction in cycle time compared to the original operational setup, validating the effectiveness of the proposed methodology for the joint optimization of space and trajectory.

3.2. Case Study 2: Evolutionary Optimization in a Robotic Assembly Cell

The RC dedicated to the assembly process consists of five workstations arranged around an industrial robot. The central station (Cell #3) is where the assembly takes place, while the remaining stations hold the parts to be assembled. Once the process is completed, the finished product is removed by an external system.

Table 5 shows the dimensions of each workstation, which serve as reference data for applying the proposed optimization methodology. The total dimensions of the cell are 1305 mm in length and

 doi:10.20944/preprints202507.0847.v1

1180 mm in width, resulting in a total operational area of 1.54 m². The exclusion zone of the robot base has a radius of 395 mm.

Table 5. Dimensions of Robotic Assembly Cell.

Stations	length (mm)	Width (mm)
Assembly 3	750	400
Station 1	350	325
Station 2	400	150
Station 4	275	225
Station 5	260	197
Robot base	395	395

This RC is equipped with a Mitsubishi Electric RV-7FRD industrial robot (Mitsubishi Electric, Tokyo, Japan), a six-axis vertical-type model installed on the floor. This robot provides full coverage around its base, as it has a 240° operating range in both directions, allowing it to easily access workstations distributed throughout its surroundings. It features a radial reach of 713.4 mm, a maximum operating speed of 11,064 mm/s, and a repeatability of ±0.02 mm.

The technical specifications of the robotic arm, along with an image of the robot installed within the cell, are shown in Figure 9.

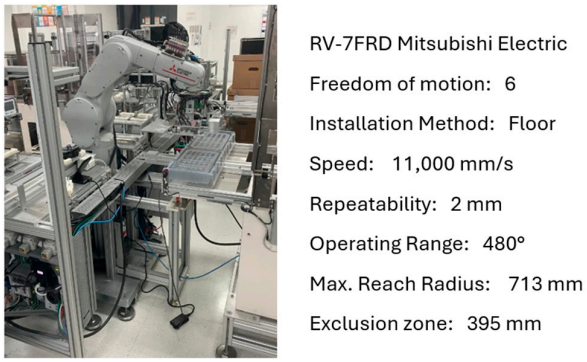


Figure 9. Mitsubishi Electric RV-7FRD robot installed in the RC with its main technical specifications.

Figure 10 shows the current layout of the RC used for the assembly process. In this configuration, the workstations are arranged in a circular pattern around the robot's base, efficiently leveraging its radial reach to cover the entire operating area.

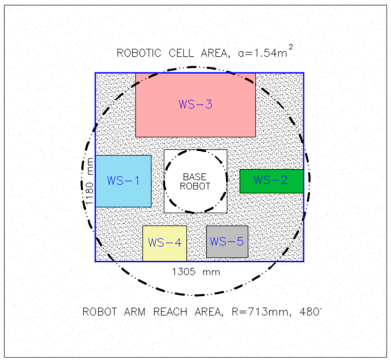


Figure 10. Current layout of the robotic assembly cell prior to applying the optimization.

Once the GA was executed, a more compact layout was obtained, featuring a linear arrangement that reorganizes the stations into a narrower strip. The new dimensions were 950 mm in width and 1080 mm in length, resulting in a total area of 1.03 m², which represents a 33.12% reduction compared to the original design. Figure 11 shows the layout generated by the GA for the assembly RC.

 doi:10.20944/preprints202507.0847.v1

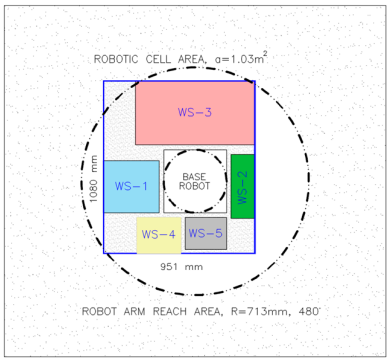


Figure 11. Optimized layout of the assembly RC obtained using the GA.

As in the previous case study, a cycle time comparison was carried out across three scenarios: (1) the current operational time, (2) the estimated time using Mitsubishi RT TOOLBOX3 PRO software with the new layout optimized by the GA, and (3) the optimized time resulting from the application of the ACO algorithm. Table 6 summarizes the results obtained, allowing for a quantitative evaluation of the performance achieved in each case.

Table 6. Cycle time comparison in the assembly cell: current configuration, simulation-based estimate, and ACO result.

Trajectory	Current Transfer Time (s)	AG+MRT Transfer Time (s)	AG+ACO Transfer Time (s)
1 to 3	2.78	2.50	2.33
2 to 3	2.61	2.02	1.93
4 to 3	2.90	2.45	2.30
5 to 3	2.41	1.89	1.80
Total Time	10.70	8.86	8.36

Based on the results obtained, the total transfer time was reduced from 10.70 s in the current configuration to 8.86 s by using the layout generated by the GA and simulated through the Mitsubishi RT TOOLBOX3 PRO software, representing a 17.20% improvement. Subsequently, the implementation of the ACO on the same spatial arrangement further reduced the total time to 8.36 s. Overall, this resulted in a 21.87% reduction in cycle time compared to the original setup.

3.3. Case Study 3: Optimal Redesign of a Robotic Screwing Cell

This RC is designed to perform a screwing operation across six workstations, which are arranged within a front-facing fixture area. Table 7 presents the dimensions of each workstation. The system uses a four-axis SCARA robot, (RH-6CRH6020-D, Mitsubishi Electric, Tokyo, Japan), which operates exclusively within the front area where the fixture is located. As a result, no rear exclusion zone is considered in this configuration. Figure 12 shows the general specifications of the robot used in this cell.

Table 7. Dimensions of Robotic Screwing Cell.

Stations	length (mm)	Width (mm)
Station 1	175	80
Station 2	170	85
Station 3	140	120
Station 4	195	180
Station 5	170	34
Station 6	120	60

 doi:10.20944/preprints202507.0847.v1

RH-6CRH6020-D Mitsubishi Electric
Freedom of motion: 4
Installation Method: Floor
Speed: 7,800 mm/s
Repeatability: 0.02 mm
Operating Range: 264°
Max. Reach Radius: 600 mm
Exclusion zone: 162.6 mm

Figure 12. Mitsubishi Electric RH-6CRH6020-D robot installed in the Robotic Screwing Cell with its main technical specifications.

Figure 13 shows the current layout of the RC dedicated to the screwing process. The operational area measures 352 mm in width and 403 mm in length, resulting in a total area of 0.142 m².

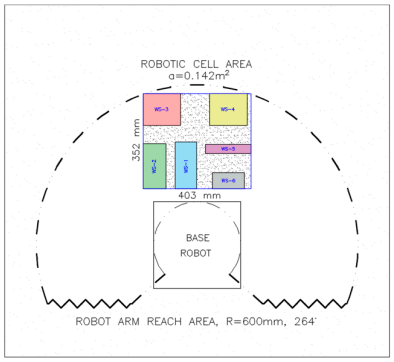


Figure 13. Current layout of the RC used for the screwing process.

Figure 14 shows the resulting configuration after applying the GA to the screwing RC. In this new layout, the workstations were redistributed to achieve a more compact and efficient arrangement. The resulting dimensions are 345 mm by 345 mm, which corresponds to a total area of 0.12 m², resulting in a 15.49% reduction in occupied area.

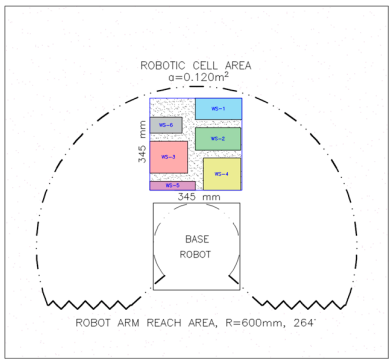


Figure 14. Optimized layout of the RC used for the screwing process.

In the first two case studies (machining and assembly), the working order of the stations was predefined. Therefore, the optimization focused solely on improving the spatial distribution of the workstations and the robot's trajectory. In contrast, in the third case (screwing), there was no predetermined sequence for visiting the workstations. This allowed the algorithm to freely modify the execution order, adding an additional layer of optimization by reorganizing both the station

 doi:10.20944/preprints202507.0847.v1

layout and the operation sequence to minimize the total cycle time. Table 8 presents the results obtained after applying the ACO to the optimized configuration of the screwing RC.

Table 8. Cycle times in the screwing cell: current configuration, GA + MRS, and GA + ACO

Actual trajectory	Current Transfer Time (s)	AG trajectory	AG+MRT Transfer Time (s)	AG+ACO Transfer Time (s)
1 to 2	1.92	5 to 3	1.78	1.47
2 to 3	2.46	3 to 6	1.98	1.64
3 to 4	3.12	6 to 1	3.41	2.81
4 to 5	2.15	1 to 2	1.85	1.53
5 to 6	1.98	2 to 4	2.10	1.81
Total Time	11.63	Total Time	9.60	9.26

In the current configuration, the total transfer time is 11.63 seconds. By applying the GA to redesign the layout and estimate the times using Mitsubishi RT TOOLBOX3 PRO software (GA+MRT), the total time was reduced to 9.60 seconds, representing a 17.45% improvement over the original configuration. An additional improvement was achieved by applying the ACO algorithm to the layout, reaching a total time of 9.26 seconds. This represents a total cycle time reduction of 20.38% compared to the original setup.

It is important to note that the new execution order of the stations contributed to shortening trajectories and generating a more logical flow within the fixture area, something that was not possible in the previous case studies where the station order was predefined.

3.3. Results Analysis

Once the proposed methodology was applied to the three case studies, a quantitative analysis was conducted to evaluate the impact of the optimization on both the utilized workspace and the total cycle time. This analysis is summarized in Table 9, which presents the values obtained for the current configurations and after the application of the GA + ACO.

Table 9. Quantitative comparison of the three case studies: area and cycle time reduction.

Case Study	Area Reduction (%)	Transfer Time Reduction with GA+MRT (%)	Transfer Time Reduction with GA+ACO (%)
Case Study 1	14.97	11.34	18.19
Case Study 2	33.12	17.20	21.87
Case Study 3	15.49	17.44	20.38
Average	21.19	15.33	20.15

In all cases, a reduction in the occupied area was achieved by spatially reorganizing the workstations, which demonstrates a significant improvement in the use of the available space. On average, a 21.19% reduction in area was obtained, which serves as a reference for the potential that the GA can offer in similar scenarios. Although this value should not be considered a universal benchmark for all RC's, it does provide a reasonable estimate of the spatial compaction capacity achievable through optimization.

Simultaneously, a substantial decrease in the robot's transfer times was observed, directly affecting the total cycle time of each cell, with an average reduction of 21.15%. This translates into greater operational efficiency, reduced bottlenecks, and increased production capacity without requiring changes to the existing hardware.

 doi:10.20944/preprints202507.0847.v1

4. Discussion

The results obtained in this study validate the effectiveness of the proposed methodology based on evolutionary algorithms for the simultaneous redesign of the workspace and the kinematic flow in industrial RC. The methodology was applied to three representative case studies, and in all three scenarios, a significant improvement was observed in key performance indicators such as occupied area and total cycle time.

One of the main strengths of the approach lies in its ability to address a multi-objective optimization problem through a sequential approximation strategy: GA reconfigures the spatial arrangement of workstations by minimizing the occupied area, while ACO optimizes the robot's trajectory over the resulting layout by minimizing transfer time. This decomposition makes it possible to efficiently address a complex combinatorial problem with nontrivial geometric and kinematic constraints. In industrial environments, where physical redesign is both costly and disruptive, this computational strategy offers a scalable, viable, and low-cost solution.

From an operational perspective, the methodology proved sufficiently robust across different topological configurations of RC, successfully adapting to circular, linear, and matrix-like layouts. Moreover, the incorporation of geometric constraints, such as exclusion zones and robot reach limitations, was effectively managed through parametric modeling of the workspace, confirming its applicability in contexts with strict spatial restrictions.

However, some inherent limitations of the current approach were identified. First, the model assumes deterministic and static operating conditions, excluding potential disruptions typical of flexible manufacturing environments, such as variable cycle times or dynamic changes in task sequencing. The integration of robust or stochastic optimization models could significantly broaden the applicability of the proposed framework. Second, the computational cost associated with the evolutionary search process, particularly in scenarios involving many stations or possible routes, can become substantial. While computational times were acceptable for the case studies presented, future applications in large-scale cells or systems involving multiple concurrent robots may require the adoption of acceleration techniques such as GPU-based parallelization, pruning heuristics, or hybrid methods incorporating machine learning.

Another critical aspect is the complexity involved in tuning the parameters of evolutionary algorithms. The performance of such algorithms is highly sensitive to the choice of parameters. As demonstrated in the sensitivity analysis, even small variations in parameter values can significantly impact the quality of the solutions obtained. This highlights the need to incorporate systematic methods for automatic parameter calibration, such as adaptive strategies, Bayesian optimization, or self-tuning mechanisms based on meta-optimization techniques.

Additionally, the initial modeling and parametrization process requires detailed technical knowledge of the robotic system and its workspace, which may hinder the adoption of the methodology in industries lacking a digital infrastructure or prior experience in advanced simulation and optimization. This underscores the importance of developing complementary tools, such as graphical interfaces and automated model validation, to facilitate its use by non-expert users.

For future research, it is proposed to extend the model to account for uncertainty in workstation availability or variability in process times, as well as to incorporate real-time sensor data to enable dynamic adaptation of the layout. Furthermore, other metaheuristic approaches, such as PSO, Differential Evolution, or hybrid algorithms integrating reinforcement learning should be explored to compare performance, convergence behavior, and robustness.

Finally, considering the growing implementation of human–robot collaborative cells, the methodology should be adapted to environments where safety, human interaction, and ergonomic constraints introduce additional restrictions on layout design and robot motion. This will be a key component in the development of more flexible, adaptive, and human-centered manufacturing systems.

 doi:10.20944/preprints202507.0847.v1

5. Conclusions

This study presented a dual optimization methodology based on evolutionary algorithms for the efficient design of industrial RC. The proposed approach sequentially and complementarily addressed two fundamental objectives: the reduction of the workspace occupied by the workstations and the minimization of cycle time through optimized robot trajectories. Experimental validation was conducted on three real-world RC with different functions (machining, assembly, and screwing), which allowed for a thorough assessment of the versatility, scalability, and effectiveness of the proposed framework.

The results demonstrated that the methodology enabled an average reduction of 21.19% in occupied area and 20.15% in operational cycle time. These percentages highlight the strong potential of computational optimization as a tool to enhance productivity in manufacturing systems, contributing to improved energy efficiency, reduced response times, and better utilization of available floor space.

Additionally, the study revealed that the predefined task sequence imposed by the process can act as a limiting factor in flow optimization. In the cases where the station order was fixed, improvements were restricted to spatial reorganization. However, in the screwing case, where the task sequence was flexible, shorter trajectories were achieved by modifying the execution order. This finding opens new opportunities for applying combinatorial algorithms in environments with high operational freedom.

Overall, this work lays the foundation for a new intelligent design paradigm for RC, aimed at continuous improvement through bio-inspired optimization tools, with immediate applications in Industry 4.0 and agile manufacturing scenarios.

Author Contributions: Conceptualization, R.-A.S.-S. and E.C.-N.; methodology, R.-A.S.-S. and E.C.-N.; software, R.-A.S.-S.; validation, R.-A.S.-S. and E.C.-N.; formal analysis, R.-A.S.-S. and E.C.-N.; investigation, R.-A.S.-S.; resources, E.C.-N.; data curation, R.-A.S.-S.; writing—original draft preparation, R.-A.S.-S.; writing—review and editing, E.C.-N.; visualization, R.-A.S.-S.; supervision, E.C.-N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed at the corresponding author.

Acknowledgments: The authors thank the CIATEQ graduate program for the support provided in carrying out this research work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Asif, S.; Bueno, M.; Ferreira, P.; Anandan, P.; Zhang, Z.; Yao, Y.; Ragunathan, G.; Tinkler, L.; Sotoodeh-Bahraini, M.; Lohse, N.; Webb, P.; Hutabarat, W.; Tiwari, A. Rapid and automated configuration of robot manufacturing cells. *Robot. Comput.-Integr. Manuf.* **2025**, *92*, 102862. https://doi.org/10.1016/j.rcim.2024.102862.
2. Bi, Z.M.; Luo, C.; Miao, Z.; Zhang, B.; Zhang, W.J.; Wang, L. Safety assurance mechanisms of collaborative robotic systems in manufacturing. *Robot. Comput.-Integr. Manuf.* **2021**, *67*, 102022. https://doi.org/10.1016/j.rcim.2020.102022.
3. Keshvarparast, A.; Battini, D.; Battaia, O.; Pirayesh, A. Collaborative robots in manufacturing and assembly systems: literature review and future research agenda. *J Intell Manuf.* **2024**, *35*, 2065–2118. https://doi.org/10.1007/s10845-023-02137-w.
4. Jiafan, Z.; Xinyu F. Challenges and key technologies in robotic cell layout design and optimization. *Proc. Inst. Mech. Eng. C: J. Mech. Eng. Sci*. **2017**. *231(15)*. 2912–2924. https://doi.org/10.1177/0954406216642473.
5. Tubaileh, A.S. Layout of robot cells based on kinematic constraints. *Int. J. Comput. Integr. Manuf.* **2015**. *28(11)*, 1142-1154, https://doi.org/10.1080/0951192X.2014.961552.

 doi:10.20944/preprints202507.0847.v1

6. Tonke, D.; Grunow, M.; Akkerman, R. Robotic-cell scheduling with pickup constraints and uncertain processing times. *IISE Trans*. **2019**. *51(11)* 1217–1235. https://doi.org/10.1080/24725854.2018.1555727.
7. Dong, J.; Pan, H.; Ye, C.; Tong, W.; Hu, J. No-wait two-stage flowshop problem with multi-task flexibility of the first machine. *Inf. Sci*. **2021**. *544*. 25-38. https://doi.org/10.1016/j.ins.2020.06.052.
8. Kats, V.; Levner, E. On the existence of dominating 6-cyclic schedules in four-machine robotic cells. *Eur. J. Oper. Res*. **2018**. *268(2)* 755–759. https://doi.org/10.1016/j.ejor.2018.01.034.
9. Sapietová, A.; Ságа, M.; Kuric, I.; Václav, S. Application of optimization algorithms for robot systems designing. *Int. J. Adv. Robot. Syst*. **2018**. *15(1)*. 1-10. https://doi.org/10.1177/1729881417754152.
10. Leiber, D.; Eickholt, D.; Vuong, A.-T.; Reinhart, G. Simulation-based layout optimization for multi-station assembly lines. *J. Intell. Manuf*. **2022**. *33*. 537–554. https://doi.org/10.1007/s10845-021-01853-5.
11. Choi, S.H.; Kim, B.S. Intelligent factory layout design framework through collaboration between optimization, simulation, and digital twin. *J Intell Manuf.* **2024**. https://doi.org/10.1007/s10845-024-02340-3.
12. Jenab, K.; Sarfaraz, A.R. A Fuzzy Graph-Based Model for Selecting Knowledge Management Tools in Innovation Processes. *Int. J. Enterp. Inf. Syst*. **2012**. *8(1)*. 1-16. https://doi.org/10.4018/jeis.2012010101.
13. Leo-Kumar, S. P. Knowledge-based expert system in manufacturing planning: state-of-the-art review. *Int. J. Prod. Res*. **2018**. *57(15–16)*, 4766–4790. https://doi.org/10.1080/00207543.2018.1424372.
14. Kuts, V.; Otto, T.; Tähemaa, T.; Bukhari, K.; Pataraia, T. Adaptive Industrial Robots Using Machine Vision. In Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition. Volume 2: Advanced Manufacturing. Pittsburgh, Pennsylvania, USA. 9–15 November 2018. https://doi.org/10.1115/IMECE2018-86720
15. Rato, D.; Oliveira, M.; Santos, V.; Gomes, M.; Sappa, A. A sensor-to-pattern calibration framework for multi-modal industrial collaborative cells. *J. Manuf. Syst*. **2022**. *64*. 497-507. https://doi.org/10.1016/j.jmsy.2022.07.006.
16. Kuts, V.; Otto, T.; Tähemaa, T.; Bondarenko, Y. Digital Twin based synchronised control and simulation of the industrial robotic cell using Virtual Reality. Journal of Machine Engineering. **2019**. *19(1)*, 128—145. https://doi.org/10.5604/01.3001.0013.0464.
17. Pérez, L.; Rodríguez-Jiménez, S.; Rodríguez, N.; Usamentiaga, R.; García, D.F. Digital Twin and Virtual Reality Based Methodology for Multi-Robot Manufacturing Cell Commissioning. *Appl. Sci*. **2020**, *10*, 3633. https://doi.org/10.3390/app10103633.
18. Vatankhah-Barenji, A.; Liu, X.; Guo, H.; Li, Z. A digital twin-driven approach towards smart manufacturing: reduced energy consumption for a robotic cell. *Int. J. Comput. Integr. Manuf*. **2020**. *34(7–8)*, 844–859. https://doi.org/10.1080/0951192X.2020.1775297.
19. Al-Salem, M.; Kharbeche, M. Throughput optimization for the Robotic Cell Problem with Controllable Processing Times. *Oper. Res*. **2017**, *51(3)*, 805–818. https://doi.org/10.1051/ro/2016064.
20. Kuratani, R.; Kojima, T.; Fujii, H.; Matoba, S.; Saitoh, Y.; Takanishi, K. Hierarchical Optimization for Robotic Cell Systems. *Trans. Inst. Syst. Control. Inf. Eng*. **2022**. *35*. 118–125. https://doi.org/10.5687/iscie.35.118.
21. Qiu, B.; Chen, S.; Gu, Y.; Zhang, C.; Yang, G. Concurrent layout and trajectory optimization for robot workcell toward energyefficient and collision-free automation. *Int. J. Adv. Manuf. Technol*. **2022**. *122*. 263–275. https://doi.org/10.1007/s00170-022-09398-4.
22. Lim, Z.Y.; Ponnambalam, S.G.; Kazuhiro, I. Nature inspired algorithms to optimize robot workcell layouts. *Appl. Soft Comput*. **2016**. *49*. 570–589. https://doi.org/10.1016/j.asoc.2016.08.048.
23. Sánchez-Sosa, R.-A.; Chavero-Navarrete, E. Robotic Cell Layout Optimization Using a Genetic Algorithm. *Appl. Sci*. **2024**, *14*, 8605. https://doi.org/10.3390/app14198605.
24. Mitsubishi Electric Corporation. RT ToolBox3 Pro. 2024. Available online: https://www.mitsubishielectric.com/fa/products/rbt/robot/smerit/rt3/index.html (accessed on 15 January 2025).
25. Feng, J.; Che, A.; Chu, C.; Levner, E.; Kats, V. Scheduling robotic cells with fixed processing times or time windows: Classification, solution approaches, polynomial algorithms and complexity. *Eur. J. Oper. Res*. **2024**. *319(2)*. 468-483. https://doi.org/10.1016/j.ejor.2024.01.041.
26. Keller, B.; Buscher, U. Single row layout models. *Eur. J. Oper. Res*. **2015**. *245(3)*. 629–644. https://doi.org/10.1016/j.ejor.2015.03.016.
27. Gen, M.; Lin, L. Genetic Algorithms and Their Applications. In: *Springer Handbook of Engineering Statistics*. Pham, H. Eds.; Springer Handbooks. Springer, London. 2023. https://doi.org/10.1007/978-1-4471-7503-2_33.
28. Alhijawi, B.; Awajan, A. Genetic algorithms: theory, genetic operators, solutions, and applications. *Evol. Intel*. **2024**. *17*, 1245–1256. https://doi.org/10.1007/s12065-023-00822-6.
29. Wu, L.; Huang, X.; Cui, J.; Liu, C.; Xiao, W. Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot. *Expert Syst. Appl*. **2023**. *215*. 119410. https://doi.org/10.1016/j.eswa.2022.119410.
30. Li, S.; Wei, Y.; Liu, X.; Zhu, H.; Yu, Z. A New Fast Ant Colony Optimization Algorithm: The Saltatory Evolution Ant Colony Optimization Algorithm. *Mathematics*. **2022**, *10*, 925. https://doi.org/10.3390/math10060925.

 doi:10.20944/preprints202507.0847.v1

31. Van Rossum, G.; Drake, F.L. Python 3 Reference Manual; CreateSpace: Scotts Valley, CA, USA, 2009. Available online: https://www.python.org/ (accessed on 25 August 2024).
32. Microsoft. Visual Studio Code. 2024. Available online: https://code.visualstudio.com/updates/v1_91 (accessed on 25 August 2024).