



Optimization Techniques for Multi Cooperative Systems MCTR 1021

Mechatronics Engineering

Faculty of Engineering and Materials Science

German University in Cairo

SWARM ROBOTS FOR DYNAMIC FIREFIGHTING (ADAPTIVE COVERAGE CONTROL) (PROGRESS REPORT)

By

Team 39

Mahmoud Ahmed

Hager Mohamed

Sarah Fahmy

Shahd Elfeky

Shahd Hesham

Course Team

Salah

November 15, 2025

This is to certify that:

- (i) the report comprises only our original work toward the course project,
- (ii) due acknowledgment has been made in the text to all other material used

Team 39
November 15, 2025

1. Introduction

This report presents our progress for Milestone 4, where we extend the optimization framework for *Swarm Robots for Dynamic Firefighting* by implementing a Genetic Algorithm (GA) and conducting a comprehensive comparison between GA and the previously developed Simulated Annealing (SA) optimizer. Our objective is to analyze the quality of solutions, computational behaviour, and robustness of both algorithms when applied to the multi-robot adaptive coverage problem. All experiments were conducted using the modular Python package developed in earlier milestones.

2. Problem Understanding

The core problem remains the same as in previous milestones: a team of mobile robots must continuously cover a dynamically evolving fire boundary. The fire grows according to an expanding and rotating elliptical model, while robots must remain within environmental, kinematic, and communication constraints.

The optimization problem seeks to minimize the weighted objective:

$$J = \alpha_1 J_{\text{cov}} + \alpha_2 J_{\text{dist}} + \alpha_3 J_{\text{bal}} + \alpha_4 J_{\text{time}},$$

subject to speed, minimum-distance, connectivity, workspace, and energy constraints. The optimizer manipulates both the velocity sequence $U \in \mathbb{R}^{2T \times N}$ and the time step Δt .

The main task in this milestone is to evaluate the performance of GA as a global-search stochastic method and compare it with SA based on objective decomposition, convergence behaviour, and constraint satisfaction.

3. Genetic Algorithm Implementation

The GA was implemented in `ga.py` following a standard evolutionary structure, adapted to the specifics of our problem.

Population Representation Each individual chromosome encodes:

$$\text{Individual} = \{U, \Delta t\},$$

where U stores all robot velocity commands and $\Delta t \in [0.20, 0.80]$ is optimized simultaneously.

Selection Tournament selection (size = 3) was chosen for its robustness and ability to maintain diversity.

Crossover A blended crossover operator (BLX- α) was applied on the continuous U and Δt values. This operator is appropriate for real-valued search spaces and allows controlled exploration around parent solutions:

$$\text{child}_i = p_i^{(1)} + \gamma(p_i^{(2)} - p_i^{(1)}), \quad \gamma \sim \mathcal{U}(-\alpha, 1 + \alpha).$$

Mutation Gaussian mutation with adaptive variance was used to encourage local exploration. Post-mutation, velocities are projected onto the feasible speed set, and Δt is clipped to its bounds.

Pseudocode

```

Initialize population P of size M with random (U, dt)
Evaluate fitness of all individuals

for generation = 1 ... G:
    Select parents using tournament selection
    Apply BLX-alpha crossover to produce offspring
    Apply Gaussian mutation to U and dt
    Project velocities and clamp dt to feasible bounds
    Evaluate offspring fitness
    Perform elitist survivor selection
end for

```

Problem-Specific Considerations

- Each evaluation requires simulating the fire model and robot kinematics.
- Penalty terms are large (10^4 scale) to ensure strict constraint satisfaction.
- The algorithm logs both total objective and objective-only values to distinguish penalized from feasible behaviour.

4. Case Studies

We reused the two scenarios from earlier milestones to evaluate the scalability of GA and compare it with SA.

Case A (Small)

- 3 robots, horizon $T = 12$
- Workspace $[0, 12] \times [0, 10]$ with one rectangular obstacle
- Fire model: rotating elliptic expansion

Case B (Full)

- 5 robots, horizon $T = 25$
- Same workspace and obstacle

Shared Optimization Parameters

$$\Delta t \in [0.20, 0.80], \quad \alpha = (1.0, 0.1, 0.05, 0.4).$$

GA Parameters

- Population size: 30
- Generations: 80
- Tournament size: 3
- Crossover rate: 0.9
- Mutation rate: 0.2

SA Parameters (for comparison)

- Initial temperature: $T_0 = 6.0$
- Cooling rate: 0.96
- Iterations: 500

5. Validation of the Genetic Algorithm

Validation was performed by verifying:

- Correct handling of constraints through penalty terms.
- Consistency of trajectory feasibility.
- Expected asymptotic decrease in objective-only convergence.

Across all simulations, GA produced feasible solutions with zero constraint violations, indicating that the penalty-based enforcement was correctly integrated. The objective-only convergence curves show clear improvement across evaluations, confirming valid optimizer behaviour.

6. Results for Case Studies

6.1 Case A (Small)

- GA converged to a feasible solution with $\Delta t = 0.20$ s, similar to SA.
- The travel-distance and time components dominated the objective.

Objective Breakdown

Method	J_{cov}	J_{dist}	J_{bal}	J_{time}
SA	1.71	5.31	0.16	2.40
GA	1.53	5.17	0.54	2.40

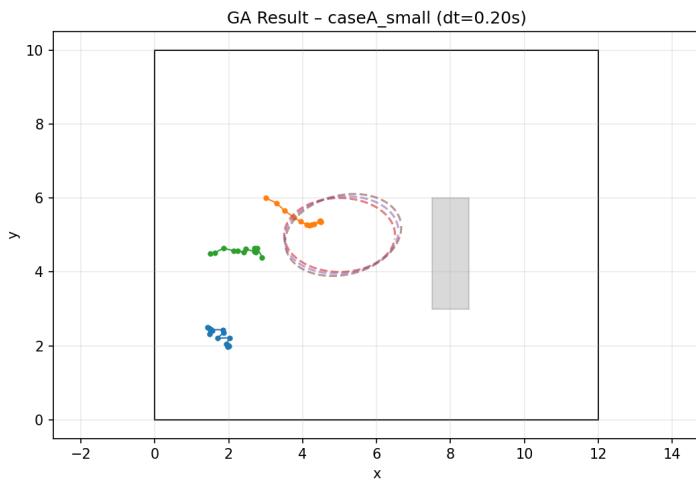


Figure 1: Case A: GA-optimized trajectories.

Trajectory Visualization (GA)

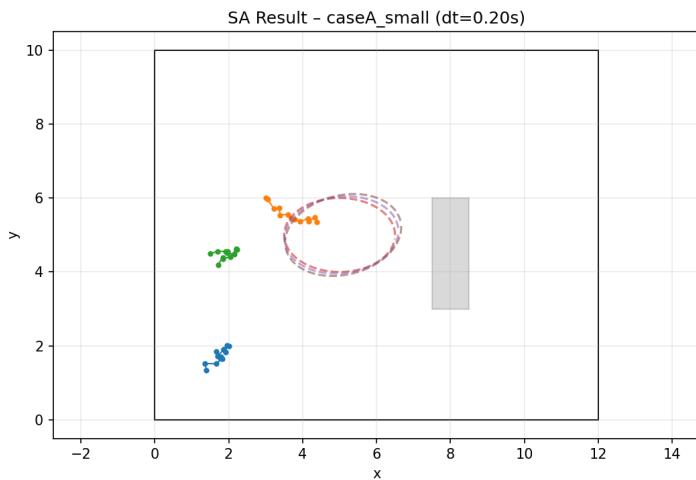


Figure 2: Case A: SA-optimized trajectories.

Trajectory Visualization (SA)

6.2 Case B (Full)

- GA required more evaluations to reach competitive performance.
- Both methods achieved zero penalty violations.

Objective Breakdown

Method	J_{cov}	J_{dist}	J_{bal}	J_{time}
SA	3.21	21.51	6.39	5.00
GA	2.95	24.30	0.63	5.00

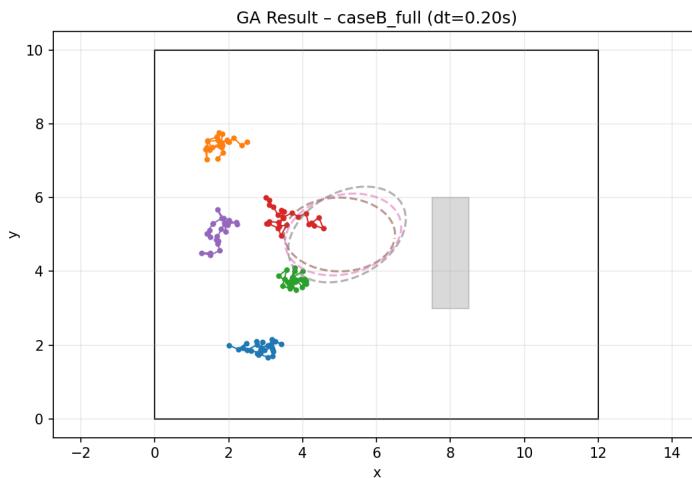


Figure 3: Case B: GA-optimized trajectories.

Trajectory Visualization (GA)

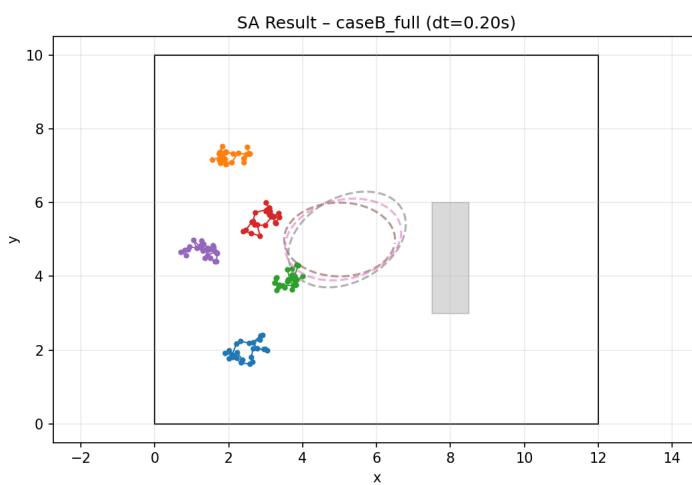


Figure 4: Case B: SA-optimized trajectories.

Trajectory Visualization (SA)

7. SA vs GA Comparison

We compared both algorithms using the following indices:

- **Coverage quality (J_{cov}):** GA slightly improved boundary matching.
- **Travel efficiency (J_{dist}):** SA achieved lower travel in Case B.
- **Energy balance (J_{bal}):** GA produced more balanced solutions.
- **Time efficiency (J_{time}):** Both selected $\Delta t = 0.20$.
- **Convergence behaviour:** SA converges faster; GA allows broader exploration.

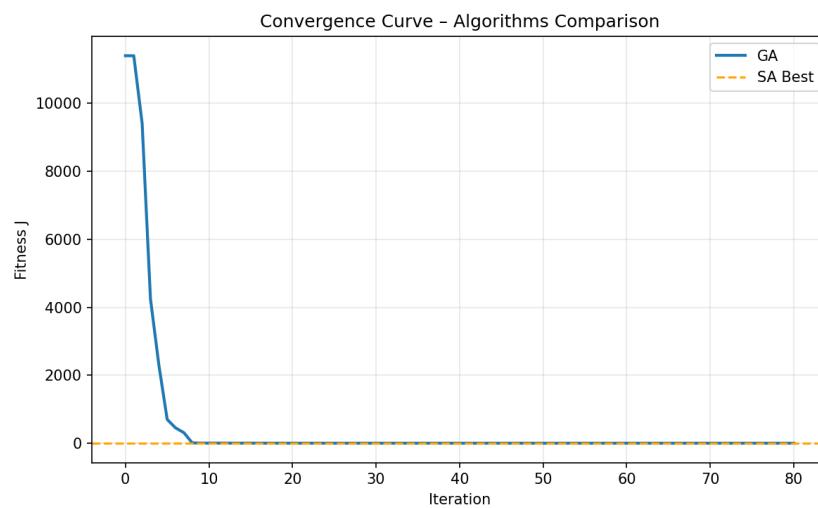


Figure 5: Case A: Total-cost convergence curves for SA and GA.

Convergence Curves

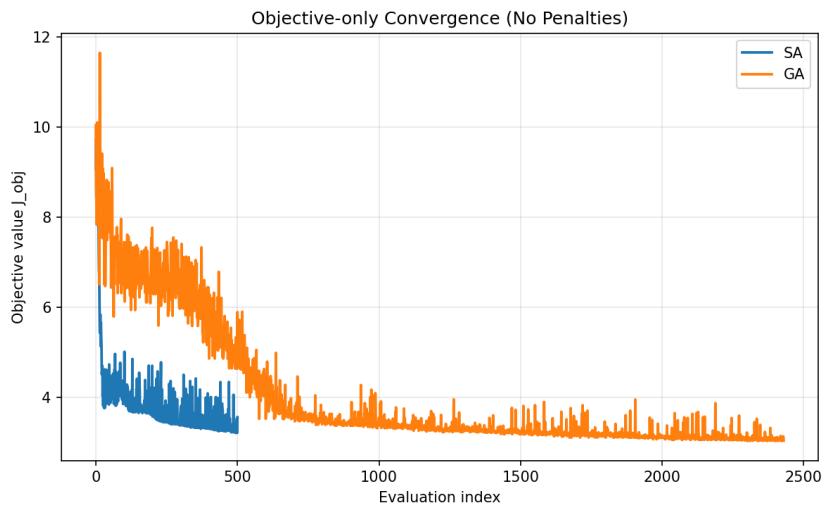


Figure 6: Case A: Objective-only convergence (no penalties).

Objective-Only Convergence

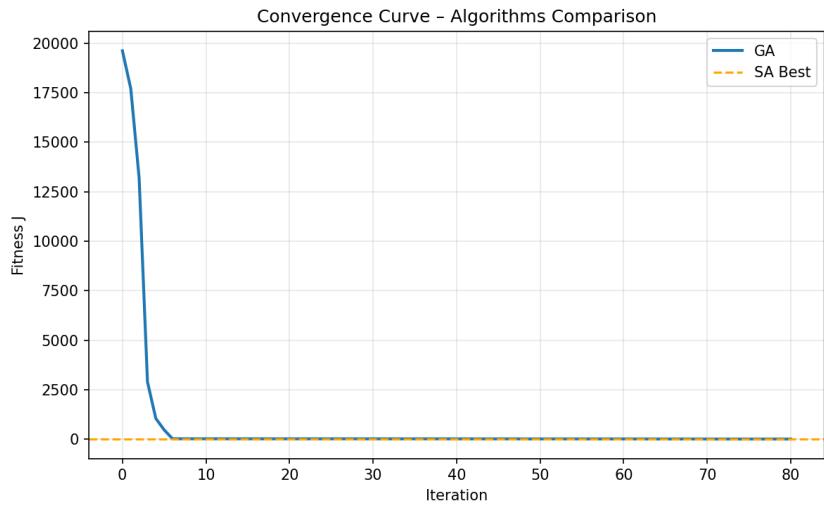


Figure 7: Case B: Total-cost convergence curves for SA and GA.

Convergence Curves (Case B)

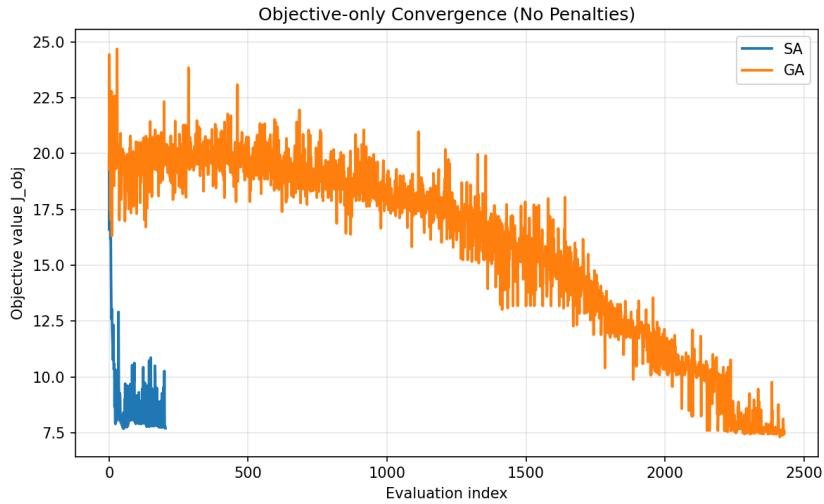


Figure 8: Case B: Objective-only convergence (no penalties).

Objective-Only Convergence (Case B)

8. Performance Analysis

Observations

- SA demonstrates rapid descent due to its local stochastic nature.
- GA shows smoother long-term improvement and better energy balance.
- Both algorithms consistently delivered feasible, constraint-satisfying solutions.
- In larger scenarios, GA required more evaluations but yielded comparable performance.
- In all experiments, both algorithms satisfied all constraints, resulting in zero penalty terms.

Effect of Parameters

- Increasing population size improves GA stability but increases computation.
- Mutation rate strongly influences GA trajectory diversity.
- SA performance is highly sensitive to cooling rate.

9. Conclusion

GA was successfully implemented and validated for the adaptive firefighting coverage problem. Both GA and SA achieved feasible and effective multi-robot trajectories across all scenarios. SA converges faster, while GA provides better global exploration and slightly improved coverage and energy balance.

10. Next Steps

- Implement Particle Swarm Optimization (PSO) for further comparison.
- Investigate adaptive mutation and hybrid metaheuristics.
- Extend testing to additional fire-spread models and larger robot teams.

11. How to Run the Code

1. Activate environment: `.\env\Scripts\Activate.ps1`
2. Run GA: `python -m swarm_fire_optimization.run_ga_demo`
3. Run SA: `python -m swarm_fire_optimization.run_sa_demo`
4. Comparison: `python compare_algorithms.py`