Real-Time Evolutionary Optimization of Infrared Receiver Parameters in Dynamic Swarm Communication

By

Jinzhe Liu

MSc Robotics Dissertation

School of Engineering Mathematics and Technology
UNIVERSITY OF BRISTOL
&
School of Engineering
UNIVERSITY OF THE WEST OF ENGLAND

A MSc dissertation submitted to the University of Bristol and the University of the West of England in accordance with the requirements of the degree of MASTER OF SCIENCE IN ROBOTICS in the Faculty of Engineering.

August 28, 2025

Declaration of own work

I declare that the work in this MSc dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Jinzhe Liu 20 Aug 2025

Ethics statement

This project did not require ethical review as determined by my supervisor.

Jinzhe Liu 20 Aug 2025

Real-Time Evolutionary Optimization of Infrared Receiver Parameters in Dynamic Swarm Communication

Jinzhe Liu

***Abstract*— This study addresses the problem of optimising infrared communication parameters in swarm robots. Infrared links in swarm robots are sensitive to receiver timing and polling. This dissertation embeds a lightweight genetic algorithm (GA) on an M5Core2 to adapt four receiver parameters in real time (rx_timeout, rx_byte_timeout, polling mode, channel index). In 100 generations (population 10), we compare selection and mutation/crossover settings under single-/dual-transmitter and dynamic-topology scenarios. Relative to a fixed baseline, on-board evolution delivers 3 times higher steady-state performance, preserves responsiveness with single sources, stabilises dual-source reception, and rapidly recovers after scheduled topology shifts. The most reliable setup uses tournament selection (k=3), 35% mutation, and 10% crossover. These results validate online evolution as a practical route to adaptive, resilient IR communication for swarm systems.**

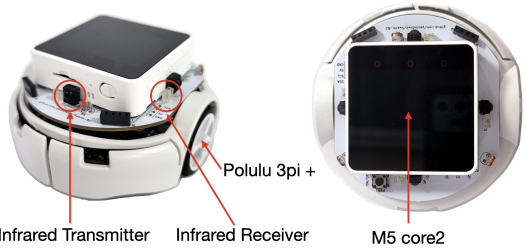

Fig. 1. The swarm robot has three layers. The Pololu 3pi+ is only used for movement.

I. Introduction

In recent years, swarm robotics has been developed primarily through observations of social animals such as ants and bees, leading to robotic systems exhibiting swarm intelligence characteristics similar to those of these species [1][2]. Swarm robots rely on multiple agents cooperating to accomplish tasks, with the core requirement being the timely and reliable exchange of information. Among various communication methods, infrared (IR) communication is widely used for short-range data exchange between small mobile robots due to its low cost, low power consumption, and strong physical locality [3][4][5]. Therefore, reliable infrared communication is one of the main objectives of swarm robots.

Infrared communication is attractive for localised, low-cost links but remains highly sensitive to transceiver configuration. Trenkwalder et al. proposed *SwarmCom*, an infrared-based self-organising network that dynamically adjusts its reception parameters based on thresholds [4]. Although our hardware platform focuses on improving information reception efficiency rather than signal strength, this approach improves robustness and the probability of error-free transmission. Therefore, we focus on optimising receiver timing parameters that can be adjusted directly in real time.

For real-time optimisation of infrared receiver parameters, other work has explored the use of a genetic algorithm (GA) in local communication systems [6]. For instance, Liu et al. proposed an azimuth estimation module based on multiple IR sensors [7], optimised sensor weights, and used an eight-IR-direction Mona robot to improve spatial layout via a GA. This significantly enhanced the accuracy of relative orientation estimation, validating the effectiveness of GAs in multi-variable optimisation and emphasising the importance of jointly tuning hardware layout and software parameters. Their results demonstrate the feasibility of applying GAs to IR communication optimisation on resource-constrained swarm robotic platforms.

Although prior work has revealed the effects of orientation, occlusion, and communication settings on IR channels in real platforms, the systematic evaluation of onboard, online evolutionary optimisation for dynamic IR parameters remains under-explored. This represents a promising opportunity to enhance the reliability and adaptability of IR communication in complex and dynamic scenarios for swarm robots' movement and configuration.

Bredeche proved how robots evaluate their fitness through interactions in real environments, rather than relying on pre-simulated models[8]. In particular, the system can adaptively adjust its behaviour and parameters in response to changes in the number of robots, topological reorganisations, and environmental perturbations. Compared to manual configuration or fixed parameters, the adaptive nature of evolution highlights its importance as a path to achieving long-term autonomy and adaptability in swarm robotics. Existing simulation-based optimisation methods often overestimate performance because they cannot capture the random and dynamic characteristics of real-world communication. Therefore, in our real-world experiments, our achievement is for good scalability, robustness, and population diversity to facilitate reliable and efficient message exchange.

To this end, we were inspired by Liu's work on azimuth [7] and Norouzi's work on optimising wireless sensor networks using GA [9] and employ a GA running directly on the robots' embedded controllers, which is hypothesised to enhance communication performance by adapting key receiver parameters online. Therefore, GA is guided by a

fitness function that rewards successful message reception and activity.

The key contribution of this GA lies in its ability to operate online and optimise in real time: (1) all parameters (in parameters for optimisation) can be dynamically reconfigured during operation without device reboot or manual intervention; (2) All evaluations are performed directly on hardware, thereby avoiding performance overestimation associated with simulation-only studies. This architecture (hardware and firmware layer) not only demonstrates the feasibility of adaptive IR communication in swarm robots but also provides a scalable framework for increasing the efficiency of the communication in swarm robots.

Research Aims:

- To investigate whether online genetic algorithms can enhance the reliability and adaptability of infrared communication in swarm robotic systems.
- To understand how dynamic parameter optimisation affects communication performance under varying swarm topologies and environmental disturbances.

Research Objectives:

- Design and implement an embedded GA framework supporting real-time evaluation and parameter updates.
- Construct a multi-robot IR communication experimental system to collect GA's performance data under various parameter configurations.
- Analyse the GA's evolutionary process, optimisation of different mutation rates and crossover rates, and contribution to communication stability.

To validate the proposed approach on real hardware, we were inspired by Talamali's experiment on how swarms of robots can better adapt to changes in dynamic environments when they have shorter communication ranges or fewer communication links [10]. We design and conducted four studies spanning static and dynamic conditions: (i) a baseline without GA to establish a fixed-parameter reference; (ii) a single-transmitter scenario to assess point-to-point reliability and convergence speed; (iii) a dual-transmitter scenario to test robustness under concurrent sources and increased interference; and (iv) a dynamic-topology scenario where transmitters were relocated and added/removed during execution to emulate changing neighbour orientations and group composition. In addition, we compared two different GA selection strategies: roulette-wheel and tournament to examine their impact on convergence speed, diversity preservation, and robustness under the same receiver parameters. We hypothesize that online evolution will (H1) outperform fixed baselines after convergence, (H2) favour shorter timeouts under single-source conditions to preserve responsiveness, (H3) in multi-source conditions, converge to a longer receive window and enable polling to reduce information loss, and (H4) reconfigure parameters rapidly after topology changes to maintain a stable fitness.

Across these studies, the embedded GA consistently exceeded the fixed baseline and exhibited the predicted adaptations: higher message success rate in single-source operation and greater stability with dual-source operation, while maintaining reliable performance in relocation and addition/removal transmitter situation(dynamic).

Through these efforts, this research not only validates the feasibility of online evolutionary optimisation in multi-robot communication but also provides an experimental foundation and methodological reference for deploying adaptive communication mechanisms in future swarm robotic systems.

II. System Implementation

A. System Overview

The real-time optimisation system (Fig.1) is based on an M5core2 microcontroller platform to drive a four-direction infrared board via I2C. The system was developed by Paul [11].

The M5Core2 is based on an ESP32 running at 240 MHz, serves as the primary embedded controller, supporting GA computation and data buffering. The IR communication module connects via I2C, allowing dynamic parameter tuning and status reading. It features a low-power IR LED and a high-sensitivity IR receiver with a reception covering the standard 38 kHz modulation frequency that mitigates the effects of ambient light. These parameters in Table I from the IR receiver directly affect packet reception success rate and latency, and thus form the core variables optimised by the GA.

TABLE I
RECEIVER PARAMETERS OPTIMIZED BY THE GENETIC ALGORITHM

Parameter name	Type
`rx_timeout`	uint16_t
`rx_byte_timeout`	uint8_t
`rx_cycle`	float (0–1, thresholded to bool)
`rx_pwr_index`	float (0–1, quantized to 0–3)

In this study, the system was used as a platform to investigate how online genetic algorithms (GAs) adapt to communication parameters under actual hardware and interference conditions. The focus of this study was to evaluate the performance of GAs and understand their contribution to the reliability and adaptability of swarm robot communication. The system consists of the hardware layer and firmware layer.

- Hardware Layer: On the hardware side (M5Core2), the embedded GA periodically reconfigures the receiver's communication parameters(See II-B). Transmission and reception settings are written via the I2C interface to an external IR module, enabling dynamic adjustment and state monitoring during each fixed evaluation window.
- Firmware Layer: An Arduino Nano is mounted on the IR communication module to control IR transmission. It communicates with the M5Core2 via I2C for parameter updates and data exchange. UART frames are generated at 4800 bps and then modulated onto a 38 kHz IR carrier for transmission[11]; demodulated signals are decoded back into UART bytes on the receiver, and calculate CRC when sending and verify CRC when receiving.

This multi-layer configuration ensures that the hardware layer influences the configurable parameters for message reception, while the firmware layer provides the control and data exchange required for real-time application and testing of these parameters. Together, they ensure that each evolutionary step is evaluated on an actual embedded platform, making the research both realistic and scientifically relevant, rather than purely simulated.

B. Parameters for Optimisation

The performance of infrared (IR) communication in swarm robotic systems is influenced by several receiver parameters. The following subsections describe the function and mechanism of each parameter in detail, highlighting their impact on communication performance and their role within the optimisation process.

In our setup, the parameters in Table I are controlled by the system, and can be directly reconfigured at runtime through I2C by sending a message. To enable adaptive optimisation using a genetic algorithm (GA), we focus on four key parameters: `rx_timeout`, `rx_byte_timeout`, `rx_cycle`, and `rx_pwr_index`. They jointly determine the robustness of message reception, the latency of communication, and the spatial coverage of the sensing system.

1) rx_cycle (Boolean): Channel polling (periodic) switch. When enabled, the receiver cycles through the 4 channels using a "stay-and-switch" scheme. When disabled, it continuously listens on a single fixed channel.

- *Periodic mode (ON):* The receiver polls its 4 infrared sensors in sequence. The total listen time per polling cycle is given by `rx_timeout` and is evenly divided across channels. This increases spatial coverage and the chance of detecting neighbours from multiple directions. (For example, with 4 sensors, each channel receives 1/4 of the cycle timeout.)
- *Stationary mode (OFF):* Polling is disabled and the receiver listens continuously on one fixed channel, allocating the entire `rx_timeout` to that channel. This concentrates the message in a known direction and improves same-direction stability, at the cost of losing other orientations.

2) rx_timeout (unit: ms): Maximum waiting-time budget per *polling cycle*. When `rx_cycle` is ON, the receiver switches to the next channel after each sensor (See rx_cycle for each sensor's timeout). When `rx_cycle` is OFF, the entire `rx_timeout` applies to the selected channel, determining both the risk of missing short frames and the single-response latency.

3) rx_byte_timeout (unit: ms): The maximum allowable time interval between two consecutive bytes within the same frame. If this interval exceeds the threshold, the current frame is considered interrupted or timed out, and further reception is aborted.

This parameter determines tolerance to jitter and non-uniform byte intervals:

- *If set smaller than suitable value:* In the presence of interference or processing delays, the transmitter may not complete sending all byte data before the threshold is reached, causing the receiver to misinterpret the situation as a frame interruption, leading to dropped frames.
- *If set bigger than suitable value:* The receiver may occupy the channel for an extended time while waiting for the next byte, reducing effective throughput and polling efficiency, and slowing overall reception.

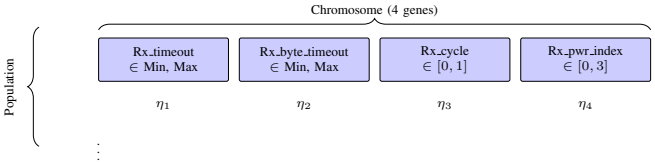

Fig. 2. Chromosome for the range of each parameter

4) rx_pwr_index (Discrete, 0–3): Specifies the sensor channel to be used in a given direction. When `rx_cycle` is `OFF`, this parameter determines the fixed channel for continuous listening. When polling is `ON`, this parameter is ignored.

In fixed listening mode($rx\_cycle = off$), selecting the channel facing the communication partner or the side with less interference can improve success rate and signal-to-noise ratio.

C. GA Implementation

In contrast to many existing studies that evaluate genetic algorithms in simulation or high-performance computing environments [12], our implementation executes directly on embedded hardware in real time. A lightweight embedded implementation realises all stages of the GA—fitness evaluation, population initialisation, selection (we use two selection operators (Roulette wheel and tournament)), crossover, mutation, termination, and elitism. Within this framework, each encodes four genes that represent receiver configurations (See Fig.2 and Sec.II-B). These genes are hypothesised to influence system robustness, latency, and spatial coverage; accordingly, the fitness function defines the criteria for evaluating communication quality.

D. Fitness Function

The fitness function defines the optimisation objective of the genetic algorithm and serves as the fundamental measure of communication quality. Each robot is equipped with multiple IR receivers arranged around its body to provide omnidirectional coverage. Assume we are in periodic mode ($rx\_cycle = on$), the firmware polls these receivers sequentially, meaning that the effective communication reliability depends not only on the robustness of a single channel but also on how well the system maintains coverage across different spatial directions. If optimisation were to focus solely on successful packet reception, the GA might converge prematurely to configurations that perform well in a single direction but ignore others. To counter this, we introduce channel activity as a secondary factor-while packet reception

directly measures reliable data exchange, channel activity provides an indirect indication of whether a receiver is exposed to potential transmissions in its direction. In this experiment, the fitness function consists of two components, representing the *reception success count* and the *communication efficiency* of the receiver. The fitness calculation formula is expressed as:

$$\text{fitness} = \sum_{i=1}^{4} (X \cdot \text{pass\_count}_i + Y \cdot \text{activity}_i) \quad (1)$$

Where:

- pass_count_i — Number of successfully received packets on sensor i.
- activity_i — Activity level of channel i, representing signal detection activity.
- The coefficient X assigns a high weight to packet reception success, making it the primary optimisation objective.
- The coefficient Y gives a lower weight to channel activity, encouraging exploration of both periodic and stationary modes while prioritising reliable reception.

(1) The successful packet reception (pass count) is defined as the event in which a complete data frame is correctly received and decoded by the infrared communication board. For each valid packet received during an evaluation window, the metric is incremented by one. Thus, the value represents the total number of successfully received packets, directly reflecting the reliability of the communication link.

(2) Activity represents the signal activity level detected by each infrared receiver within the evaluation window. It includes not only successfully received frames but also incomplete or corrupted ones, and thus serves as an indirect measure of channel occupancy.

In our implementation, the coefficients were empirically set to $X = 100$ and $Y = 5$. This weighting scheme reflects the design choice that successful packet reception is the primary optimisation objective. A high $X : Y$ ratio ensures that the GA prioritises reliable communication, whereas a small but non-zero Y prevents the algorithm from converging prematurely to configurations that succeed only in a limited spatial direction. The chosen values were validated through preliminary experiments, which confirmed that to demonstrate that this parameter can achieve high fitness while preventing the GA from prematurely converging to a configuration that only works in a single direction.

E. Population Size

In our implementation, the population size was fixed at 10. On the embedded platform, every individual must be evaluated directly on the hardware, meaning that larger populations (e.g., 20–50 individuals) would increase the evaluation time per generation and reduce the system's real-time responsiveness. Conversely, using smaller populations (fewer than 10 individuals) would reduce genetic diversity and cause premature convergence. Preliminary experiments confirmed that a population size of 10 provides a suitable trade-off between evaluation cost and population diversity: it ensures lower evaluation time but can maintain high fitness.

F. Population Initialization

During the initialisation phase, the values of all genes were assigned using a uniform random distribution. At this stage, both the fitness values and evaluation timers were reset for each individual. Such uniform random initialisation ensures high diversity in the initial population, thereby promoting broad exploration of the search space at the early stages of evolution. The specific value ranges for each parameter are not detailed here, as they are systematically defined and discussed in the Experimental Methodology section (Sec.III-B.1).

G. Selection

The selection operator chooses individuals for crossover and mutation. This study discusses two strategies for choosing the best selection method for the experiment:

Roulette Wheel Selection: The selection probability is proportional to individual fitness. Although this favours highly fit individuals, it also risks premature convergence if extreme fitness values dominate the population. In our experiments, this issue is particularly relevant because swarm robots operate in dynamic scenarios: transient conditions (e.g., a temporary alignment of robots or a momentary reduction in interference) can yield abnormally high fitness values. If such outliers are disproportionately amplified through selection, the GA may converge prematurely to parameter configurations that are only effective under those temporary conditions, thereby reducing long-term adaptability.

Tournament Selection: The tournament selection is a random subset of k individuals drawn from the current population, and the fittest individual within this subset is chosen as a parent candidate. This method is less sensitive to noisy or inaccurate fitness evaluations. However, the choice of k directly affects the balance between selection pressure and population diversity: larger tournament sizes increase selection pressure but risk premature convergence, whereas smaller tournament sizes maintain diversity but may slow convergence. In our implementation, the tournament size was fixed at $k = 3$. For each parent selection, three individuals are sampled uniformly at random from the current population, and the fittest among them is chosen. This tournament is repeated independently each time a parent is needed, until the full set of parents required to produce the next generation of 10 individuals is obtained.

H. Elitism

Elitism directly copies the current global best individual into the next generation, ensuring that the optimal configuration is not lost during reproduction. In our implementation, the elitism size was set to 4. This value corresponds to approximately 40% of the total population ($N = 10$). While large elitism fractions can exacerbate the "extreme fitness dominates" effect by reducing diversity, in our experiments, this value provided a workable compromise. The remaining

60% of the population was still subject to variation through crossover and mutation, which ensured continued exploration of the parameter space.

A smaller elitism size (e.g., one or two individuals) was observed in preliminary tests to risk losing promising traits due to stochastic variation, whereas larger values reduced population diversity and led to premature convergence. The choice of four elites, therefore, reflects a compromise, ensuring stability of the evolutionary process while still allowing new genetic material to propagate.

I. Crossover

Crossover combines partial genetic information from two parents to produce potentially fitter offspring. With probability p_c, two parents $P^{(1)}$ and $P^{(2)}$ generate an offspring C by recombination; otherwise, one parent is copied:

$$C = \begin{cases} \text{CROSSOVER}(P^{(1)}, P^{(2)}), & \text{with prob. } p_c, \\ P^{(q)},\ q \sim \text{Bernoulli}(0.5), & \text{with prob. } 1 - p_c. \end{cases}$$

For integer-type genes (`rx_timeout`, `rx_byte_timeout`), a single-point crossover is applied at one of the predefined cut positions, while categorical/continuous genes (`rx_cycle`, `rx_pwr_index`) follow uniform inheritance, where the offspring copies the value from either parent with equal probability. This design preserves parameter coherence while maintaining diversity and mimics natural genetic recombination. Separating the treatment of integer-type and categorical/continuous genes ensures evolutionary stability: integer parameters benefit from single-point crossover to maintain coherent value ranges, while categorical/continuous parameters are inherited uniformly to avoid infeasible or unstable intermediate states.

J. Mutation

Mutation maintains population diversity by introducing stochastic perturbations and thus prevents premature convergence. In this work, per-gene mutation is applied with parameter-specific probabilities in Table.I: for each gene g, mutation occurs with probability $p_m(g)$, yielding.

$$g' = \begin{cases} g + \Delta_g, & \text{with probability } p_m(g), \\ g, & \text{otherwise,} \end{cases}$$

Where Δ_g is a bounded random perturbation. The process can be interpreted as follows:

- Integer-type genes (`rx_timeout`, `rx_byte_timeout`) mutate by adding a discrete offset within predefined limits (see III-B. The maximum step size for `rx_timeout` is

$$R = \frac{\texttt{MAX} - \texttt{MIN}}{10}, \tag{2}$$

  i.e., about 10% of the total range. Thus, `rx_timeout` mutates within [20,1000] ms, and `rx_byte_timeout` mutates within [1,5] ms. This allows us to find the best fitness in a smaller range.

- Float-type genes (`rx_cycle`, `rx_pwr_index`) are perturbed by adding uniform noise $U(-0.5, 0.5)$ and clipped to $[0, 1]$. When applied to hardware, `rx_cycle` is thresholded at 0.5 (ON if $\geq$ 0.5, OFF otherwise), while `rx_pwr_index` is quantised into $\{0, 1, 2, 3\}$ based on quartiles $[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1]$.

Boundary constraints ensure all mutated values remain within valid ranges, preventing invalid hardware configurations.

III. Experimental Methodology

To evaluate the effectiveness of the proposed genetic algorithm (GA) under different communication scenarios, a set of comparative experiments was conducted. The motivation for these experiments is twofold. First, communication in swarm robotic systems is inherently dynamic and subject to disturbances such as interference, varying numbers of transmitters, and changing spatial configurations. Second, while the GA is designed to adapt parameters online, its practical benefit must be demonstrated across conditions that mirror real swarm deployments.

Therefore, these experiments are essential to explain the motivation:

- establish a baseline performance without GA for comparison to prove the benefit of GA;
- verify that the GA can improve communication reliability and anti-interference ability under stable conditions;
- demonstrates adaptability and diversity in dynamic environments where swarm topology changes during operation.

Together, these experimental methods provide indicative evidence that the proposed GA can enhance IR communication reliability in our experimental setups, while also maintaining a degree of robustness and adaptability in multi-robot scenarios. However, these findings should be interpreted within the constraints of the tested hardware and experimental conditions, and further studies are needed to assess scalability and generality in larger swarms or more complex environments.

A. Experimental Setup

The experiments employed three identical M5Core2 with an infrared (IR) transceiver. One device acted as the receiver running the GA optimiser, which dynamically adjusted communication parameters (see II-B) and reported fitness values via serial output. Transmitters periodically sent standardised packets (`''test''`) as the signal source. This message only takes 8 bytes (4-byte message + 4-byte checksum); it can be easier to calculate the ideal fitness and avoid using more evaluation time because the sent data is too long.

Environment: Robots were positioned at a fixed distance of 5 cm to avoid signal saturation or excessive attenuation. Physical barriers ensured that each receiver channel only detected signals from its intended direction.

Each experiment ran for 100 generations, with every individual evaluated over a 1000-ms window. While generation count provides a convenient measure of evolutionary progress, it does not directly correspond to real elapsed time on hardware. Results are therefore reported in terms of generations, which also provides a hardware-agnostic view of convergence speed. Transmitters used fixed settings of `tx_period = 100 ms` and `tx_repeat = 5`, which were chosen to approximate realistic sending behaviour in swarm robotic systems. Based on "test" message and Sec.III-A.1.a, this configuration should provide sufficiently frequent transmissions to represent practical multi-robot interactions, and the repetition factor may help to mitigate packet loss and ambient disturbances, although these effects were not explicitly quantified in our experiments.

1) Evaluation Metrics:

a) Ideal Fitness/reference bound: To establish an analytical upper bound, the ideal fitness was derived based on transmission timing and message structure of the adopted hardware. At a bitrate of 4800 bps, each byte consists of 10 bits, requiring:

$$T_{\text{byte}} = \frac{10}{4800} \approx 2.08\,\text{ms}. \tag{3}$$

A complete test message of 8 bytes (Include 4 bytes for start, end, and checksum, implemented in [11]). Therefore requires:

$$T_{\text{msg}} = 8 \times T_{\text{byte}} \approx 16.7\,\text{ms}. \tag{4}$$

In our firmware, $tx\_repeat = 5$ denotes five back-to-back retransmissions within one send (no 100 ms spacing between repeats), one repeated transmission occupies:

$$T_{\text{msg\_rep}} = 5 \times T_{\text{msg}} \approx 83.3\,\text{ms}. \tag{5}$$

Including the fixed transmission period of 100 ms, the effective cycle duration becomes:

$$T_{\text{cycle}} = T_{\text{msg\_rep}} + 100 \approx 183.3\,\text{ms}. \tag{6}$$

Within an evaluation window of 1000 ms, the number of complete cycles is:

$$N_{\text{cycles}} = \left\lfloor \frac{1000}{183.3} \right\rfloor = 5\ period. \tag{7}$$

Each cycle delivers 5 packets, yielding 25 packets in total. The residual time at the end of the window allows approximately 5 additional packets, giving:

$$\text{pass\_count}_{\max} = 5\ message * 5\ period + 5 = 30. \tag{8}$$

In ideal conditions, channel activity is a complete lossless reception. Since activity is proportional to the number of detected messages, activity was assumed to match the maximum pass count (30) in ideal conditions.

Finally, the theoretical maximum fitness is expressed as:

$$\text{fitness}_{ideal} = X \cdot \text{pass\_count}_{\max} + Y \cdot \text{activity}_{best}. \tag{9}$$

For the chosen weights $X = 100$ and $Y = 5$, this evaluates to:

$$\text{fitness}_{ideal} = (100 + 5) \times 30 = 3150. \tag{10}$$

It should be noted that, when the observed fitness exceeds the theoretical maximum, this does not imply that the system received more data than physically possible. Instead, it arises from the contribution of the activity term, which assigns weight to partially received or corrupted signals as well. This design encourages the GA to explore diverse parameter settings rather than converging prematurely to a single stable channel. Hence, fitness values above the theoretical bound are expected and reflect the exploratory role of activity, not a violation of physical limits.

B. Justification of Parameter Settings

1) Timeout Determination: The value of `rx_timeout` was determined based on a combination of hardware constraints, performance trade-offs, and empirical considerations. According to the theoretical fitness derivation, each byte requires approximately 2.08 ms for transmission. Therefore, the reception window must cover an entire byte sequence to avoid frame truncation. Thus, $rx\_timeout_{\text{minimum}}$ should be above $16.7 \approx 20\ ms$ (Allow more time). A larger timeout (e.g., 500–1000 ms) increases the likelihood of capturing weak signals and introduces longer waiting time with no message sent during omnidirectional polling.

In this study, the $rx\_timeout$ range was set to 20–1000 ms, and the GA was employed to evolve the optimal values under varying interference conditions. Preliminary experiments indicated that $rx\_byte\_timeout$ should be assigned a smaller value under static transmission conditions, whereas a larger range is required to accommodate transmitter mobility. Based on this observation, the parameter was fixed accordingly in Table II.

2) Crossover and Mutation Rates: These value was chosen before experimentation and constrained within a reasonable range. In theory, crossover and mutation serve to maintain population diversity and prevent premature convergence[13], [14], [15]. Due to the dynamic environments in this experiment, we appropriately increased the range of consideration for the two values, providing more possibilities to explore more parameters by increasing the crossover rate and mutation rate. Thus, this study tested the selection operator, crossover rate of 10-50% (in 20% increments), and mutation rates of 25–45% (in 10% increments), to test if ensuring population diversity while maintaining convergence speed.

C. Comparison Groups

To assess GA-based optimisation comprehensively, the following scenarios were tested with evaluation and parameter updates occurring once per generation.(See Table II):

- **Phase 1: Operator and parameter study.** In the first phase, different selection operators (roulette-wheel and tournament) were compared by running 100 generations, and multiple crossover/mutation rates were tested. The goal was to identify crossover and mutation configurations that provide a balance between convergence speed and population diversity.

TABLE II

GA EXPERIMENTAL CONFIGURATIONS.

Name	Crossover/Mutation	rx_timeout	byte_timeout
	Phase I		
	Selection		
Roulette	10% / 25%	20–1000	1–5
Tournament	10% / 25%	20–1000	1–5
	Mutation Rate		
Low	10% / 25%	20–1000	1–5
Medium	10% / 35%	20–1000	1–5
High	10% / 45%	20–1000	1–5
	Crossover Rate		
Low	10% / 25%	20–1000	1–5
Medium	30% / 25%	20–1000	1–5
High	50% / 25%	20–1000	1–5
	Phase II		
Baseline	(no GA)	20	1
Single	Defined by the first phase	20–1000	1–5
Dual	Defined by the first phase	20–1000	1–5
Dynamic	Defined by the first phase	20–1000	1–10

- **Phase 2: Communication experiments.** Using the operator and parameter settings determined in Phase 1, to further validate the stability of GA optimisation, we conducted the same generation (100) experiments and a 5-point moving average was applied to the plotted curves to reduce visual fluctuations. This smoothing is purely for presentation and does not affect the underlying experimental results:

  a) Baseline: Receiver operated under fixed parameters without GA (point-to-point).

  b) Single transmitter: A point-to-point setup tested GA optimisation under stable communication.

  c) Dual transmitters: Two concurrent transmitters simulated multi-robot communication.

  d) Dynamic scenario: To emulate changing swarm topologies, transmitters were dynamically relocated or added during execution. Specifically, at the beginning, we put one transmitter pointed at channel 3. At generation 10, one transmitter was physically repositioned so that its beam was directed toward the receiver's channel 2 sensor instead of channel 3, altering the receiver's relative orientation. There are 20 generations here to see if it has completely shifted to the new direction. At generation 30, the transmitter was returned to its original position(channel 3). At generation 40, a second transmitter was introduced to simulate concurrent multi-source communication, and at generation 50, one transmitter was removed, returning the system to a single-source configuration. These scheduled changes created controlled disturbances in the communication environment, thereby testing the adaptability of the genetic algorithm to dynamic and unpredictable swarm conditions.

In summary, the experimental methods and parameter settings are detailed in the Table II

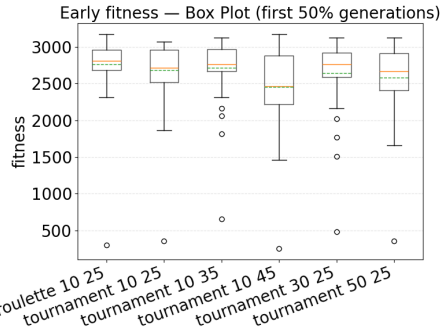

Fig. 3. Early-state fitness distributions (First 50% generations) under different GA configurations.The x-axis information represents selection, crossover rate, and mutation rate. The green line is the mean value. The orange line is the median.

IV. EXPERIMENTAL RESULTS

To systematically evaluate the impact of genetic algorithm (GA) parameters, we conducted experiments based on the configurations listed in Table II. The focus was placed on

- Phase 1: the effects of crossover rate, selection operator, and mutation rate, with results presented in terms of both evolutionary dynamics and steady-state performance.
- Phase 2: assessing GA adaptability under varying communication conditions, including single- and multi-transmitter scenarios as well as controlled dynamic disturbances (e.g., transmitter relocation). This phase highlights how the GA responds to environmental changes and tests its robustness in realistic swarm settings.

It is worth noting that values exceeding the ideal fit do not necessarily mean that the system receives values that exceed what is physically possible(Detail in Sec.III-A.1.a). At the same time, because the sender's transmission time and the receiver's reception time are not synchronised, the receiver may sometimes miss information even with the best settings. Therefore, even when the fitness is converging, there will be fluctuations. Therefore, when the fluctuations in the fitness become smaller, we can judge that it has converged.

A. Phase 1: compare the effects of crossover rate, selection operator, and mutation rate.

1) Effect of Selection Operator: Figure 3 presents the distribution of early-stage fitness values (first 50% of generations) across different GA configurations. Overall, the median fitness mostly falls within the range of 2500–3000, indicating that all settings achieved relatively strong performance in the initial evolutionary period.

Roulette selection produced comparatively stable outcomes, with a narrower interquartile range and fewer low outliers, suggesting reliable convergence behaviour in early generations. By contrast, tournament selection exhibited higher variance across configurations: although median values were similar to those of roulette, the wider spread and presence of extremely low outliers indicate greater stochastic variability. These observations suggest that roulette selection favours

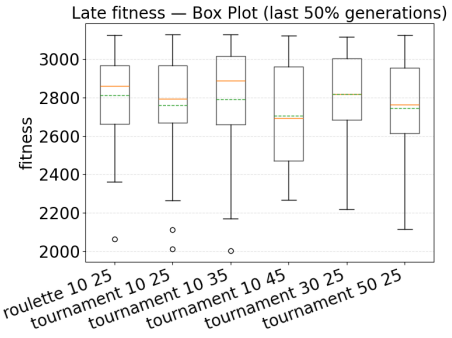

Fig. 4. late fitness distributions (Last 50% generations) under different GA configurations. Legend information is the same as 3. Compared with the early fitness, the late fitness has fewer outliers and smaller fluctuations, showing a convergence trend(steady-state).

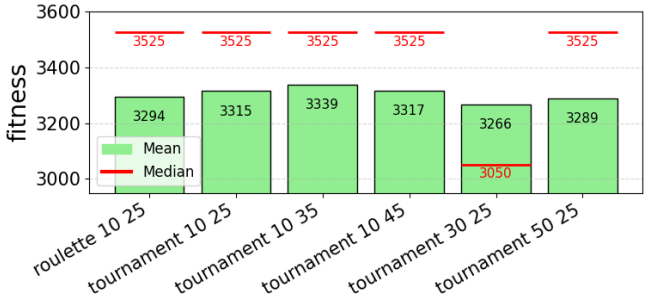

Fig. 5. Average best fitness under different GA parameter configurations.

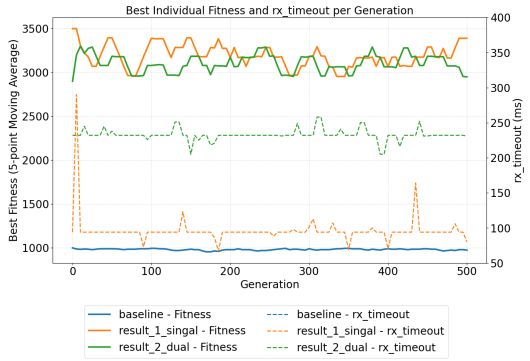

Fig. 6. Best individual fitness and $rx\_timeout$ across generations (5-point moving average) under baseline(timeout 20 ms), single-transmitter, and dual-transmitter configurations.

stable convergence in the early stages, whereas tournament selection promotes broader exploration at the cost of increased variance in fitness outcomes.

Fig.4 tests compared roulette-wheel and tournament selection for the last 50% generation. Both achieved similar steady-state fitness. Our experimental requirements demand greater diversity due to the dynamic nature of the experiment. Therefore, even though the two models have similar fitness, and Roulette even outperforms them, the tournament model more consistently preserves population diversity. Therefore, based on the two graphs, we use tournament selection (size = 3) as the default operator in all subsequent experiments.

2) Effect of Crossover Rate: We evaluated three crossover probabilities: 10%, 30%, and 50% base on Fig.3 and 4. The results indicate the following findings.

Low crossover (10%): In the early stage, 10% shows higher medians and a tighter spread than 30% (and often than 50%). In the steady state, its medians remain among the highest with comparable or lower variance. Lower recombination pressure likely preserves building blocks and avoids oscillations. Overall, 10% offers the best balance between early-stage performance, stability, and steady-state level.

Moderate crossover (30%): Across our runs, 30% exhibits the widest IQR and lower early-stage medians, and its steady-state medians are consistently below those of 10%. Under non-stationary conditions, this setting is more fluctuation-prone, so its advantage is not consistent.

High crossover (50%): While 50% sometimes reaches competitive levels, it generally yields higher variance and more low-end outliers across phases, with a greater tendency to oscillate mid-to-late.

Overall, considering early responsiveness, robustness to hardware noise, and steady-state performance, we adopt crossover = 10% as the default crossover rate (with tournament selection).

3) Effect of Mutation Rate: The analysis of the mutation rate is similar to that of the crossover rate, so it can be referred to crossover rate. In conclusion, experiments with different mutation probabilities revealed that moderate levels (25–35%) achieved the most reliable performance.

As shown in Fig.3 - 5, mutation rates of 25–35% achieved higher medians with relatively narrow distributions, indicating a good balance of exploration and convergence. At 45%, the distribution widened substantially with more outliers, reflecting excessive randomness.

4) Combined Effect of Mutation and Crossover: Figure 5 summarises the average best fitness achieved by each configuration. All GA-based optimisations achieved average best fitness values (pass_count + activity) between 3266 and 3339, compared to the lower baseline (Fig.6). The highest average fitness (3339) was achieved under a configuration combining tournament selection with a mutation rate of 35% and a crossover rate of 10%. This is consistent with our previous judgment. Accordingly, this setting was adopted as the main configuration in subsequent experiments.

B. Phase 2: Comparison of Baseline, Single and Dual Transmitter Configurations

Figure 6 shows the best individual fitness per generation. The baseline configuration is low (around 1000) (parameter in Table II), achieving only about one-third of the theoretical ideal fitness (3150). By contrast, both results fluctuated around the theoretical ideal fitness. In the single-transmitter, peak fitness values reached around 3400, slightly above the theoretical bound. In the dual-transmitter case, fitness values stabilised around 3100–3300, corresponding to 98–105% of the ideal, and exhibited reduced variance compared to the single-transmitter setup. This demonstrates that GA optimisation is effective across both single- and multi-source

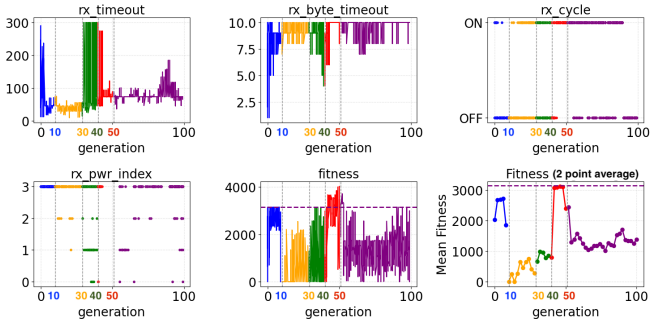

Fig. 7. Evolution of receiver parameters and fitness under dynamic transmitter relocation and addition/removal. Colours denote topology phases. The purple horizontal line indicates the ideal fitness (3150), not a hard upper bound; values above it reflect activity contributions. (see III-A.1.a)

conditions, with the dual-transmitter configuration favouring stability over peak performance.

Fig. 6 presents the evolution of the `rx_timeout` parameter. The baseline remained fixed at 20 ms, while GA optimisation yielded distinct adaptive behaviours:

In single-source conditions, the timeout consistently converges to shorter values, increasing polling frequency and responsiveness without sacrificing delivery rate. Under dual sources, it evolves longer, which preserves frame integrity per direction and reduces variance, yielding more stable steady-state fitness.

These results further confirm that GA can autonomously adjust communication parameters to match the transmitter topology. In particular, the algorithm balances fast refresh rates in single-source conditions with extended reception times in multi-source conditions, thereby maintaining robustness across diverse swarm communication scenarios.

C. Adaptability in Dynamic Scenarios

Fig. 7 illustrates the evolution of receiver parameters across these dynamic phases:

- Generations 0–10 (blue): The system initially relied on channel 3. Rapid parameter adjustments were observed, leading to an immediate increase in fitness as the algorithm quickly adapted to the environment.
- Generations 10–30 (yellow): After relocating the transmitter to channel 2, the `rx_pwr_index` distribution shifted accordingly, with a few individuals converging on channel 2. However, some residual values persisted on channel 3, indicating that the population had not fully converged and continued to explore alternative configurations.
- Generations 30–40 (green): Once the transmitter was returned to channel 3, fitness improved again. At the same time, `rx_timeout` values increased, reflecting reconfiguration efforts as the algorithm adapted back to the previous communication link.
- Generations 40–50 (red): With the introduction of a second transmitter, `rx_cycle` was more frequently enabled (ON), enabling multi-channel polling. The stabilised `rx_timeout` and `rx_byte_timeout` were longer than in the single-transmitter phases, consistent with the need for extended reception windows under concurrent communication. Fitness values exceeding the reference indicate increased activity contributions.
- Generations 50–100 (purple): After reverting to a single-transmitter configuration, fitness declined from its dual-source peak but gradually stabilised. Meanwhile, `rx_pwr_index` exhibited exploration across multiple channels, reflecting attempts to re-identify the most reliable signal source.

Overall, the dynamic experiment demonstrates that the GA was able to autonomously adapt receiver parameters in response to sudden and repeated topology changes. The observed patterns, such as extending timeouts under dual-transmitter conditions, highlight the robustness of GA-driven parameter optimisation in dynamic and unpredictable swarm communication environments.

D. Summary of Results

Comprehensive experimental results demonstrate the feasibility and advantages of deploying an onboard genetic algorithm (GA) for swarm robot communication. The algorithm consistently outperforms a fixed baseline in various configurations. The GA rapidly improves fitness in the early stages of evolution and maintains superior performance in the subsequent steady-state phase. Regarding parameter settings, while roulette wheel selection may perform slightly better in some cases, tournament selection, due to its robustness to skewed distributions and controllable selection pressure, is a more practical choice for embedded applications. A crossover rate of 10% and a mutation rate of 35% were shown to achieve the optimal balance between convergence speed, system stability, and final fitness. Furthermore, experiments with transmitter topologies demonstrated the GA's adaptive capabilities: it autonomously adjusts parameters to accommodate single or dual transmitters and rapidly recovers performance after dynamic changes such as transmitter movement, addition, or removal, demonstrating strong robustness. These findings lay the foundation for building more scalable and resilient swarm systems.

V. Discussion and Conclusion

A. Key Findings

This study demonstrates that embedding a genetic algorithm (GA) in resource-constrained swarm robotic platforms enables real-time optimisation of infrared (IR) communication parameters. The key findings are as follows:

- **Enhanced reliability and adaptability:** The GA autonomously tuned receiver parameters according to the transmitter setup.
  (1) With a single transmitter, the algorithm converged to shorter timeouts (About 90–100 ms), improving responsiveness and polling frequency. (2) Under dual transmitters, the GA evolved towards longer timeouts (About 230 ms) and more frequent polling, enhancing frame integrity and reducing variance. This adaptive behaviour validates the hypothesis that online evolution

can tailor communication strategies to environmental conditions.

- **Adaptability in dynamic environments:** When communication topology changed during execution, GA autonomously adjusted receiver parameters, maintaining stable communication without manual intervention.
- **Balanced parameter settings:** Moderate mutation rates (25–35%) and crossover rates (around 10%) achieved the best balance between convergence speed and population diversity, consistent with theoretical expectations in evolutionary computation.

B. Comparison with Existing Work

Recent reviews indicate that automated design and evolution methods in swarm robotics are still dominated by simulation or offline optimisation, which further exacerbates the "reality gap" between simulated performance and physical system behaviour. Kuckling systematically reviewed advancements in offline evolution, embodied/on-board evolution, and swarm learning, explicitly stating that simulation-led evaluation remains mainstream[16]. He advocated for evaluation on real robots to directly confront common deployment-scenario challenges such as transmission uncertainty, environmental noise, and resource constraints. Consistent with this, our research directly deploys a genetic algorithm on an embedded microcontroller for online, real-time optimisation under a dynamic topology and a real infrared link. The focus is on tuning receiver-side timing and polling parameters, and we validate the improvements in reliability (successful packets) and communication efficiency compared to a fixed baseline, even in the presence of multi-transmitter interference.

C. Limitations

Despite the promising results, several limitations remain:

- **Scale:** Experiments were conducted on only three robots, and scalability to larger collectives has not yet been validated.
- **Limited Communication:** The current study focuses on point-to-point short message transmission, with messages limited to an 8-byte (4-byte message + 4-byte checksum)"test" string. In real swarm tasks, longer message exchanges are often required. Such more complex communication patterns may lead to longer evaluation times and other challenges, and it remains unverified whether the GA can still adapt efficiently. The fitness function and the set of parameters subject to GA optimisation may need to be redesigned.
- **Idealised Experimental Environment:** In the experiments, robots were positioned at a fixed distance of 5 cm, with physical barriers used to prevent crosstalk. Although the evaluation was performed on real hardware, the environment was still overly controlled.

  In more realistic swarm deployments, robot distances and orientations change more dynamically, while occlusion and even reflection can affect IR communication performance. As a result, the reported findings may overestimate the adaptability and robustness of the GA.

D. Future Work

Future research will focus on several promising directions. First, we should go beyond short point-to-point messages and evaluate GA-based optimisation under longer and more complex message exchanges. This will require a reconsideration of both the fitness function design and the set of parameters under evolutionary control, to capture issues such as throughput, buffering delays, and multi-packet integrity. Second, we will investigate scalability to larger robot populations that will enable the evaluation of convergence behaviour and stability when multiple simultaneous communication links compete for limited receive sensors.

REFERENCES

[1] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal, "Kilobot: A low cost robot with scalable operations designed for collective behaviors," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 966–975, Jul. 2014.

[2] M. Dorigo, G. Theraulaz, and V. Trianni, "Swarm robotics: Past, present, and future [point of view]," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, Jul. 2021.

[3] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," entry incomplete: add venue/year or replace with an official publication.

[4] S. M. Trenkwalder, I. Esnaola, Y. K. Lopes, A. Kolling, and R. Groß, "Swarmcom: an infra-red-based mobile ad-hoc network for severely constrained robots," *Autonomous Robots*, vol. 44, no. 1, pp. 93–114, 2020.

[5] F. Arvin, K. Samsudin, and A. R. Ramli, "Development of ir-based short-range communication techniques for swarm robot applications," *Advances in Electrical and Computer Engineering*, vol. 10, no. 4, pp. 61–68, 2010.

[6] Álvaro Gutiérrez, E. Tuci, and A. Campo, "Evolution of neuro-controllers for robots' alignment using local communication," *International Journal of Advanced Robotic Systems*, vol. 6, no. 1, p. 6, Mar. 2009.

[7] Z. Liu, C. West, B. Lennox, and F. Arvin, "Local bearing estimation for a swarm of low-cost miniature robots," *Sensors*, vol. 20, no. 11, p. 3308, 2020.

[8] N. Bredeche, E. Haasdijk, and A. Prieto, "Embodied evolution in collective robotics: A review," *Frontiers in Robotics and AI*, vol. 5, p. 12, 2018.

[9] A. Norouzi and A. H. Zaim, "Genetic algorithm application in optimization of wireless sensor networks," *The Scientific World Journal*, vol. 2014, p. 286575, 2014.

[10] M. S. Talamali, A. Saha, J. A. R. Marshall, and A. Reina, "When less is more: Robot swarms adapt better to changes with constrained communication," *Science Robotics*, vol. 6, no. 56, p. eabf1416, Jul. 2021.

[11] P. O'Dowd, "paulodowd/swarm-b2," https://github.com/paulodowd/Swarm-B2, 2025, gitHub repository, accessed Aug. 19, 2025.

[12] J. C. Bongard, "Evolutionary robotics," *Communications of the ACM*, vol. 56, no. 8, p. 74, 2013.

[13] A. E. Eiben and J. E. Smith, "Evolutionary robotics," in *Introduction to Evolutionary Computing*, A. E. Eiben and J. E. Smith, Eds. Berlin, Heidelberg: Springer, 2015, pp. 245–258.

[14] K. Deb and T. Goel, "Controlled elitist non-dominated sorting genetic algorithms for better convergence," in *Proc. Evolutionary Multi-Criterion Optimization (EMO 2001)*, ser. Lecture Notes in Computer Science, E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, Eds., vol. 1993. Berlin, Heidelberg: Springer, 2001, pp. 67–81.

[15] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, Apr. 2015.

[16] J. Kuckling, "Recent trends in robot learning and evolution for swarm robotics," *Frontiers in Robotics and AI*, vol. 10, p. 1134841, 2023.