JSST
Journal of Space Science and Technology

P-ISSN: 2008-4560
E-ISSN: 2423-4516

Journal of Space Science and Technology

Vol. 18, No. 4, pp. 48-62, 2025
https://doi.org/10.22034/jsst.2025.1555
Journal Homepage: https://jsstpub.com

Original Research Paper

Optimization of Genetic Algorithm Parameters for the Design of Indigenous Telecommunication Satellite Constellations

Arash Kosari [1*] and Amir Reza Fathi[2]

1. Department of Electrical Engineering and Information Technology, Iranian Research Organization for Science and Technology (IROST), Tehran, Iran
2. University of Tehran, Tehran, Iran

ARTICLE INFO

Article History:

Received 27 June 2025
Revised 22 September 2025
Accepted 07 October 2025
Available Online 15 October 2025

Keywords:

Satellite constellations
Optimization
Regional coverage
Genetic algorithm
Indigenous constellation

ABSTRACT

The design of optimal low earth orbit (LEO) constellations is a complex, high-dimensional challenge central to modern telecommunications. While genetic algorithms (GAs) are potent optimization tools, their performance is notoriously sensitive to initial parameter settings, often leading to premature convergence and suboptimal designs. This paper introduces a novel fully adaptive genetic algorithm (AGA), engineered to overcome this critical limitation through a dynamic, self-correcting framework. The AGA employs an external control loop that co-adjusts population size and crossover fraction based on real-time metrics of population diversity and convergence stagnation. To rigorously test its resilience, the AGA was intentionally initialized with the worst-performing parameters from a static GA baseline, a configuration guaranteed to fail. First, its robustness was validated on the high-dimensional Rastrigin benchmark function, where, despite the poor start, it outperformed the best static GA by an astonishing factor of 380, proving its exceptional ability to escape deep local optima. The validated AGA was then applied to its primary mission: optimizing a 10-satellite, 30-variable LEO constellation for maximal coverage over Tehran. The results were decisive. The AGA achieved 19.45 hours of daily coverage, a 14.4% improvement over the 17.00 hours achieved by the best-tuned static GA. More remarkably, it accomplished this in just 15 hours, an 80.8% reduction from the 78 hours required by the static GA. This research not only contributes a highly efficient and reliable methodology for indigenous satellite constellation design but also presents a powerful paradigm for solving complex engineering problems where parameter sensitivity is a major bottleneck.

* Corresponding Author's E-mail: a.kosari@irost.ir

How to Cite this Article:
A. Kosari and A. R. Fathi, "Optimization of genetic algorithm parameters for the design of indigenous telecommunication satellite constellations," *Journal of Space Science and Technology*, Vol. 18, No. 4, pp. 48-62, 2025, https://doi.org/10.22034/jsst.2025.1555.

1. INTRODUCTION

Satellite constellations have become indispensable components of modern space infrastructure, enabling critical applications such as global communication, Earth observation, navigation, and scientific research [1]. Among these, Low Earth Orbit (LEO) constellations have gained significant prominence due to their lower deployment costs, reduced signal latency, and enhanced coverage capabilities [2]. However, the design and optimization of LEO satellite constellations remain inherently complex, as they require consideration of a vast array of interdependent parameters, including satellite positions, coverage performance, link budgets, and cost efficiency. To address this complexity, the application of advanced optimization techniques has become essential. Evolutionary algorithms, and in particular, genetic algorithms (GAs), have been extensively employed for such problems due to their ability to effectively explore large, multidimensional solution spaces and avoid premature convergence to local optima [3]. Numerous studies have demonstrated the efficacy of GAs in optimizing satellite constellation configurations. For instance, Liu et al. proposed an improved GA for cost-efficient LEO constellation design [4], while Wang applied GAs to optimize satellite launch schedules, significantly enhancing mission efficiency [5]. Similarly, Guo incorporated interference constraints into GA based constellation optimization models, improving communication quality [6]. Despite these advancements, a critical challenge persists in the precise adjustment of GA parameters—namely, mutation rate, crossover rate, and initial population size—which substantially influence the algorithm's convergence behavior and solution quality [7]. Sensitivity analyses conducted by Yuan have underscored the profound impact of these parameters on LEO constellation optimization performance [8]. To address these issues, Zhao introduced adaptive parameter control mechanisms that dynamically adjust GA parameters during execution, enhancing robustness and convergence [9]. Nevertheless, most existing research focuses on general GA applications, with limited emphasis on systematic parameter optimization specifically tailored to LEO constellation design. Moreover, there is a notable research gap regarding the optimization of GA parameters for indigenous satellite constellation projects, particularly within the context of Iran's growing space capabilities. In parallel, alternative metaheuristic algorithms such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) have been explored for satellite constellation optimization, offering distinct advantages in specific scenarios [10]. These approaches have shown promise in enhancing algorithmic robustness and convergence. However, many existing adaptive methods are either overly simplistic or focus on tuning a single parameter in isolation, failing to address the intricate, synergistic interplay between them. Furthermore, there remains a notable research gap in applying and validating these advanced methods specifically for the indigenous design of complex LEO constellations, such as for Iran's burgeoning space program. This paper confronts the parameter uncertainty problem head-on by proposing a more holistic methodological solution: a novel, Fully Adaptive Genetic Algorithm (AGA). This work builds upon the promise of adaptive strategies by introducing a sophisticated control loop that co-adapts multiple key parameters in real-time, based on population diversity and convergence progress. The core innovation lies in creating an algorithm designed not just to solve the constellation design problem, but to thrive and self-correct even when initialized with non-optimal parameters. The primary contribution of this research is twofold: (1) it presents a powerful AGA methodology that significantly mitigates the critical issue of parameter sensitivity in metaheuristic optimization, and (2) it validates this methodology on a complex, 30-variable optimization problem for an indigenous LEO telecommunication constellation. By systematically moving beyond static parameter rules and demonstrating a more intelligent, self-regulating optimization process, this study provides a scientifically grounded and reproducible framework for designing the next generation of efficient and reliable satellite systems.

2. METHODOLOGY

2.1. orbital parameters

In orbital mechanics, orbital parameters constitute a set of numerical values that completely define the trajectory of a celestial object in space. These parameters are crucial for the precise calculation of a satellite's position and velocity at

any given time, the design and control of space missions, and the understanding of the dynamics governing the motion of objects within planetary systems. This discussion specifically addresses the orbital parameters of a circular orbit, a special case of an elliptical orbit, which is of particular importance due to its simplicity and diverse applications. A circular orbit is characterized by the celestial object moving around a central body (such as Earth) in a circular path. The distance between the orbiting object and the center of the central body remains constant, and the orbital speed is also uniform.

This orbit, considered the simplest form of orbit in orbital mechanics, is commonly used in initial analyses and practical applications. The complete description of a circular orbit requires the following orbital parameters:

- Semi-major Axis (a) or Orbital Radius: In a circular orbit, the semi-major axis (a) simplifies to the orbital radius, representing the constant distance of the satellite from the center of the central body. This parameter specifies the size of the orbit. Its unit is meters (m) or kilometers (km).
- Orbital Period (T): The orbital period is the duration required for the satellite to complete one revolution around the central body. This parameter depends directly on the orbital radius and the mass of the central body.
- Inclination (i): The inclination is the angle between the orbital plane and the equatorial plane of the central body (e.g., Earth). This parameter defines the orientation of the orbit relative to the equatorial plane and is measured in degrees. In a circular orbit, this parameter remains constant. The inclination determines whether the satellite will pass over high, mid, or low latitude regions.
- Right ascension of the ascending node (Ω): This parameter is the angle measured from a reference direction in the equatorial plane to the point where the orbit crosses the equatorial plane going from south to north (ascending node). It specifies the orientation of the orbit within the equatorial plane. The right ascension of the ascending node is typically measured with respect to the celestial coordinate system and is expressed in degrees. In a circular orbit, this parameter is constant.
- Argument of Periapsis (ω): This is the angle between the ascending node and the periapsis (the closest point of the orbit to the central body). In a circular orbit, the periapsis is not conventionally defined (because the distance is constant); thus, the value of this parameter is meaningless for a circular orbit. Consequently, the argument of periapsis is either undefined or conventionally considered zero for circular orbits.
- True Anomaly (ν): This parameter defines the current position of the satellite in its orbit relative to the periapsis (or a reference point). However, because the periapsis is undefined in a circular orbit, the true anomaly represents the angle between the current position of the satellite and the ascending node. The true anomaly changes with time, indicating the instantaneous position of the satellite within the orbit, and is measured in degrees.

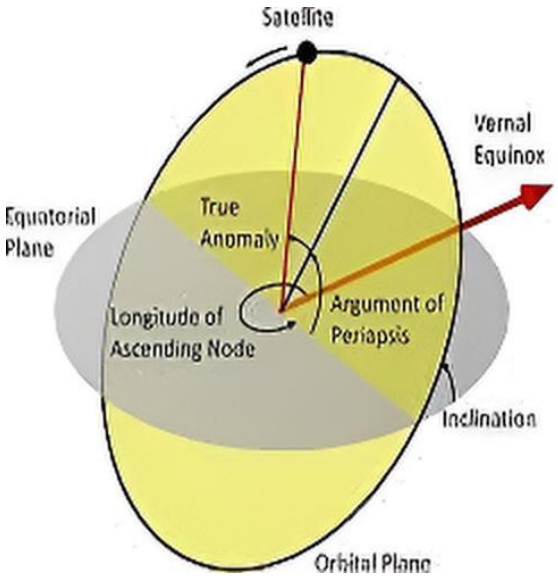

Fig. 1. Orbital elements.

Compared to elliptical orbits, the specific features of a circular orbit are simplified.

Eccentricity (e): For a circular orbit, the eccentricity is zero (e=0), indicating that the orbit is a perfect circle. The set of orbital planes of circular orbits is essential for understanding and analyzing the motion of satellites and other space objects. Accurate recognition of these maps is essential for the design, control, and analysis of space missions. Given these factors, the position and velocity of

satellites can be accurately predicted, and space missions can be successfully carried out. In this discussion, the orbital planes of a circular orbit are defined and examined in detail.

2.2. Extraction of the Physical Equations

- Simplified Specifically:

Orbital Radius: The orbital radius for a circular orbit is equivalent to the semi-major axis:

$$r = a \quad (1)$$

Orbital Period (T): The orbital period (T) can be calculated using Kepler's third law of planetary motion, which relates the period to the semi-major axis (or orbital radius in the case of a circular orbit):

$$T = 2\pi * sqrt(r^3/\mu) \quad (2)$$

where μ is the standard gravitational parameter (μ = GM, where G is the gravitational constant and M is the mass of the central body).

Orbital Velocity (v): For a circular orbit, the orbital velocity is constant:

$$v = sqrt(\mu/r) \quad (3)$$

The orbital frame, a satellite-centered coordinate system, is used to define the satellite's position and velocity. It is defined by the following axes:

- x-axis: Points in the direction of periapsis (this axis can be oriented in any direction in the orbital plane for a circular orbit)
- y-axis: Is perpendicular to the x-axis in the orbital plane in the direction of motion.
- z-axis: Perpendicular to the orbital plane, completing the right-handed coordinate system.

Position (r_orbit) in the orbital frame: In a circular orbit, the position in the orbital frame is given by:

$$r\_orbit = r\ [cos(v), sin(v), 0] \quad (4)$$

where r is the orbital radius and v is the true anomaly.

Velocity (v_orbit) in the orbital frame: The velocity in the orbital frame is given by:

$$v\_\ \text{orbit} = V\ [-\sin(\nu), \cos(V), 0] \quad (5)$$

where v is the orbital velocity.

The inertial frame, such as the Earth-centered inertial (ECI) frame, is a fixed reference frame. To transform position and velocity from the orbital frame to the inertial frame, a rotation matrix is required. This matrix is defined by three Euler rotation angles. The rotations are as follows:

- Rotation about the z-axis by the angle Ω
- Rotation about the x-axis by the angle i
- Rotation about the z-axis by the angle $(\omega + v)$

The total rotation matrix is given by the multiplication of these three rotation matrices in the following manner:

The total rotation matrix is given by the multiplication of these three rotation matrices in the following manner:

$$R = R1 * R2 * R3 \quad (6)$$

where

$$R1 = [cos(\Omega)\ -sin(\Omega)\ 0;\ sin(\Omega)\ cos(\Omega)\ 0;\ 0\ 0\ 1] \quad (7)$$

$$R2 = [1\ 0\ 0;\ 0\ cos(i)\ -sin(i);\ 0\ sin(i)\ cos(i)] \quad (8)$$

$$R3 = [cos(\omega+v)\ -sin(\omega+v)\ 0;\ sin(\omega+v)\ cos(\omega+v)\ 0;\ 0\ 0\ 1] \quad (9)$$

Note that the rotation matrices must be multiplied from right to left. Using the rotation matrix R, the position and velocity in the inertial frame, R_Inertia$_l$ and V_Inertial, are given by:

$$R\_Inertial = R * r\_orbit \quad (10)$$

$$V\_Inertial = R * v\_orbit \quad (11)$$

In this way, the speed and position of each satellite can be obtained.

2.3. Genetic Optimization Algorithm

In the context of optimization, the space encompassing all feasible solutions is termed the search space. Each point within this space represents a potential solution. Genetic Algorithms (GAs) provide a robust heuristic approach to exploring this search space in order to discover the optimal or near-optimal solutions. For example, GAs can be employed to identify the minimum of a given function. However, a fundamental challenge encountered is the presence of local minima within

the search space. These local minima can trap the GA, preventing it from reaching the global minimum. Within the framework of genetic algorithms, the optimization problem is conceptualized as an environment where potential solutions are represented as individuals populating this environment. These individuals, typically represented as binary strings or a sequence of symbols drawn from a pre-defined finite set, constitute the population in the GA. An individual in a GA is manifested in two distinct forms:

- Chromosome: This represents the raw genetic information (genotype), encoded in a manner suitable for processing by the GA. It is the fundamental structure that the GA directly manipulates through various genetic operators.
- Phenotype: This is the representation of the chromosome in terms of the actual solution in the context of the problem being addressed. It is the tangible manifestation of the genotype.

A chromosome is typically subdivided into segments known as genes. Each gene embodies a specific control factor. Each factor within the set of solutions corresponds to a gene within the chromosome. A critical aspect is that each chromosome should contain sufficient information to unambiguously define a solution within the search space. The function that transforms a chromosome (genotype) into its corresponding phenotype is typically many-to-one but must at least map one chromosome to one unique phenotype. The mapping of solutions to chromosomes does not necessarily need to be unique. In many cases, the function used for this mapping is not one-to-one. However, the mapping must at least meet the requirement of being a function; namely, every member of the domain must map to a single member of the codomain. All candidate solutions to the problem must be mapped to at least one chromosome to ensure that the entirety of the search space can potentially be explored by the algorithm. The fitness of an individual within a GA is evaluated using a fitness function, which computes the objective function for the corresponding phenotype of the individual. This function takes the phenotype as an input and provides a numerical evaluation of how well this solution fits the problem objective. To calculate fitness, the chromosome must be decoded into its corresponding phenotype, and then the fitness function is computed for that phenotype. The fitness not only indicates the suitability of the solution but also reflects how close a chromosome is to a globally optimal solution. When dealing with multi-criteria optimization problems, specifying an appropriate fitness function becomes a more complex endeavor. A major challenge is determining which solutions dominate (or are better than) other solutions.

In the ideal case, the initial population of the genetic algorithm is required to be a broad and diverse source of genes so that it is able to explore the entire search space. All possible permutations of genes need to be present in the population. Commonly, the initial population is generated randomly, ensuring a diverse gene pool from which evolution can proceed. In some cases, specific methodologies can be applied to initial population generation to aid the algorithm towards quicker convergence. However, it is crucial to ensure that the diversity is sufficiently high in the initial population. If the initial population lacks adequate diversity, the algorithm will only explore a small subset of the search space and will likely fail to converge on the globally optimal solution. The proper determination of the population size is highly dependent upon the optimization problem under analysis. Larger populations can more readily explore the search space; however, this comes at the cost of the increased computational time required for the GA to converge. It can be said that a population has converged when all individuals are very similar to each other, and further progress can be made only by using the mutation operator. Work by Goldberg has demonstrated that the efficacy of a genetic algorithm in reaching the global optima is significantly affected by the population size. The search process in a GA involves the creation of an initial population and then the generation of new populations of individuals until a pre-defined termination criterion has been met. This termination criterion can be, among others, that the population has sufficiently converged. The goal of the search process may be varied. A primary objective is to discover the optimal solution. The nature of the search algorithms that GAs use means that it is not possible to guarantee the optimality of the solution found. It is always possible that a subsequent search process can yield a better solution. In some cases, search processes might run for many years and fail

to discover a better solution than was discovered early in the run. Another objective in the search process can be faster convergence. When the fitness function evaluation requires a high computational overhead, a faster convergence is highly desirable. However, in these cases, there is a much higher chance of convergence to a local optimum or to a solution that is far away from the desired optimal solution. An alternative goal is to generate a range of distinct, but highly fit solutions. When the solution space contains a number of distinct optima with similar fitness values, the ability to produce a range of solutions becomes highly valuable since some may be more viable or have other properties making them more useful than other equally optimal solutions.

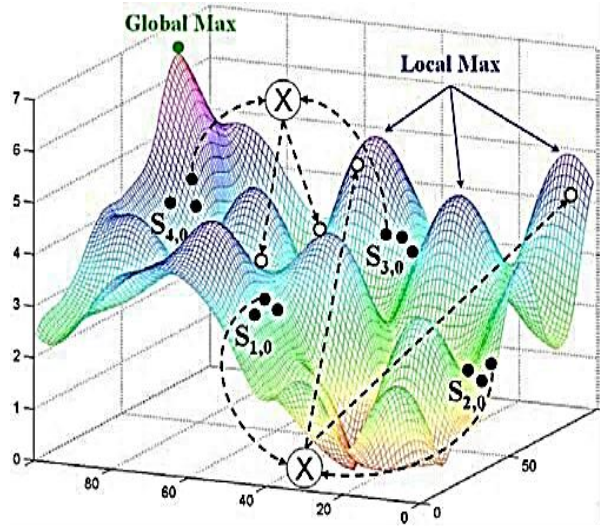

Fig. 2. Search space in the genetic optimizer.

To formulate the satellite constellation optimization problem, the design variables that the algorithm will manipulate must be defined. These variables, derived from the orbital elements described in Section 2.1, are presented in Table 1.

Table 1. Design variables.

	e	i	ω	Ω	H	θ
max	0	90	0	180	1200	180
min	0	0	0	0	1200	0
unit		deg	deg	deg	Km	deg

The primary objective of this optimization is to maximize the total access time of the satellite constellation to a designated ground station over a 24-hour simulation period. The problem is formulated as a high-dimensional optimization task, characterized by 30 design variables that define the configuration of a non-homogeneous satellite constellation. The ultimate goal is to determine the optimal set of orbital elements for each satellite that collectively yields the maximum possible coverage duration for the target ground station. The objective function, J(coverage), quantifies the total access time. It is computed by integrating a binary visibility function, $V_{const}(t)$, over the entire simulation period, T_{sim}. The visibility function is defined as:

$$V_{const}(t) = \begin{Bmatrix} 1 & \sum_{k=1}^{N_{sat}} v_k(t) \geq 1 \\ 0 & otherwise \end{Bmatrix} \quad (12)$$

where N_{sat} is the number of satellites in the constellation. The total coverage time is then given by:

$$J_{coverage} = \int_0^{T_{sim}} V_{const}(t)dt \quad (13)$$

The optimization problem can be formally stated as maximizing this objective function subject to the defined variable bounds:

$$\max_{X \in D}(J_{coverage}(X)) \quad (14)$$

where X is the 30-dimensional design vector representing the orbital elements of the constellation, and D is the feasible search space defined by the bounds in Table 1.

2.4. Robustness Analysis: Overcoming Premature Convergence

The performance of a standard genetic algorithm (GA) is critically dependent on the meticulous tuning of its operational parameters, namely population size, crossover fraction, and mutation rate. This limitation makes the standard GA a potentially unreliable tool for complex, “black-box” optimization problems where the problem landscape is unknown a priori. To overcome this critical deficiency, this research introduces a methodological innovation: a fully adaptive genetic algorithm (AGA) framework.

Instead of relying on a static, pre-determined set of parameters, our AGA framework implements a meta-heuristic control loop that dynamically monitors the state of the population during the optimization process. By analyzing key metrics such as population diversity and convergence stagnation, the framework autonomously adjusts the population size and crossover fraction between optimization epochs. This approach transforms the GA from a rigid algorithm into an intelligent, self-correcting system capable of adaptively balancing exploration and exploitation in response to the evolving search dynamics. The subsequent sections will detail the architecture of this adaptive framework and empirically validate its superior robustness and efficiency compared to the standard GA. The core innovation of this framework is an external control loop that supervises the standard GA, creating a fully adaptive genetic algorithm (AGA). This meta-heuristic layer monitors the population's state at the end of pre-defined operational intervals, or 'epochs', and dynamically adjusts the GA's core parameters for the subsequent epoch. The goal is to balance the trade-off between exploration (searching new regions of the solution space) and exploitation (refining promising solutions). This is achieved by adjusting PopulationSize and CrossoverFraction based on two key metrics: population diversity and convergence progress.

Population Diversity (D): This metric quantifies the genetic variety within the population. It is calculated as the standard deviation of the objective function scores of all individuals in the final population of an epoch. A low value indicates population convergence, signaling a high risk of being trapped in a local optimum.

$$D = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} F(x_i - \bar{f})^2} \qquad (15)$$

Here, N is the population size, $f(x_i)$ is the fitness of the i-th individual, and $\bar{f}$ is the mean fitness of the population. A small value of D indicates that most individuals have similar fitness, implying a lack of genetic diversity and a shift towards an exploitation-heavy state. Conversely, a large D suggests a diverse population actively exploring the search space.

Convergence Progress (or Stagnation): This metric tracks the algorithm's effectiveness in finding better solutions over time. It is measured by the improvement in the best fitness value from one epoch to the next. Stagnation is flagged if this improvement falls below a pre-defined stagnation-tolerance threshold S-{tol}, indicating that the current search strategy is no longer fruitful.

The adaptive control logic orchestrates the dynamic adjustment of Population Size and Crossover Fraction based on the feedback from the monitoring metrics. The rules are designed to intelligently manage the exploration-exploitation trade-off.

Triggering Exploration Mode: If the population diversity (D) drops below a diversity-threshold or if the search stagnates (improvement < S-{tol}), the algorithm infers that it may be trapped. To escape this state, it initiates an Exploration strategy.

Population Size is increased (e.g., by 25%). A larger population introduces more genetic material, broadening the search and increasing the chances of discovering new, promising areas.

Crossover Fraction is decreased (e.g., by 0.1). Reducing the crossover fraction means that a larger proportion of the next generation's individuals are created through mutation rather than by combining existing parents. This acts as a powerful mechanism to inject novelty and disrupt converged gene pools.

Triggering Exploitation Mode: If the population is sufficiently diverse and progress is being made, the algorithm switches to an exploitation strategy to efficiently refine the good solutions it has found.

Population size is decreased (e.g., by 10%). A smaller population reduces computational cost and allows the algorithm to focus its resources on the most promising individuals.

Crossover Fraction is increased (e.g., by 0.1). A higher crossover fraction promotes the combination of successful genetic traits from the best-performing parents, accelerating convergence towards the optimal solution within the current region.

In addition to the external control loop for population and crossover, our AGA leverages MATLAB's built-in adaptive mutation function, mutationadaptfeasible. Unlike a static mutation rate, this function internally adjusts the scale and direction of mutations. It operates on the principle that smaller, more focused mutations are beneficial when the population is converging (exploitation), while larger, more disruptive mutations are needed if the population stalls (exploration). The function uses the history of successful and unsuccessful

generations to determine the optimal mutation step size for the next generation. By integrating this intrinsic adaptive mutation with our extrinsic control loop for population and crossover, we create a multi-layered adaptive system that can respond to the complex dynamics of the optimization landscape at different levels.

2.4. Experimental Validation: Methodology and Benchmark Function

To empirically validate the robustness and self-correcting capabilities of our proposed adaptive genetic algorithm (AGA) framework, a comparative analysis against the standard (static) GA was designed. The primary objective of this experiment is to demonstrate the superior performance of the AGA, particularly in scenarios where the standard GA is prone to failure due to suboptimal parameter initialization.

For this purpose, we selected the Rastrigin function as the benchmark optimization problem. The Rastrigin function is a canonical and highly challenging non-convex function widely used in optimization literature to test the efficacy of global search algorithms. Its landscape is characterized by a vast number of local minima, making it extremely difficult for algorithms to locate the global minimum at *F(X)=0 for x=[0,0,…,0]* The function is defined as:

$$F(X) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i) \qquad (16)$$

For our experiment, we configured the problem with *n=12* variables, creating a complex 12-dimensional search space fraught with local optima, thus simulating the challenges encountered in our primary satellite constellation problem.

The core of our validation rests on comparing the performance of the AGA against the static GA under two distinct and revealing scenarios:

- **Scenario A**: Optimal parameter initialization (The Ideal Case): In this scenario, both algorithms are initialized with Population Size and crossover fraction values that are known a priori to be effective for the Rastrigin function. This test aims to establish a performance baseline and ensure that our AGA framework does not underperform when conditions are favorable.
- **Scenario B**: Suboptimal parameter initialization (The Realistic Challenge): This is the critical test. Both algorithms are deliberately initialized with "poor" parameter values—specifically, a small population size and a high crossover rate. This configuration is known to promote a rapid loss of genetic diversity, forcing the algorithm into premature convergence at a suboptimal local minimum. This scenario is designed to simulate a realistic situation where a practitioner, lacking full knowledge of the problem landscape, makes a poor initial judgment. The goal is to test the AGA's ability to detect this state of stagnation and autonomously correct its parameters to escape the local trap and resume the search for the global optimum.

The outcomes of these two scenarios will provide definitive evidence of the adaptive framework's value proposition: transforming the GA from a sensitive, static tool into a robust, intelligent problem-solver.

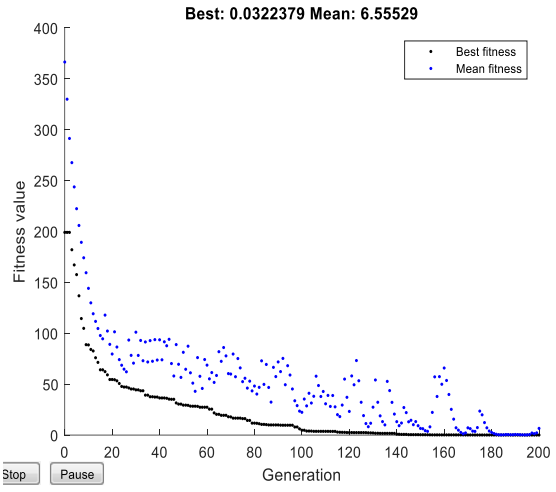

Fig. 3. Genetic optimization algorithm efforts with fixed operators and a fixed initial population of 200 and a fixed crossover rate of 0.6. (Scenario A).

The initial experiment evaluates the performance of the standard Genetic Algorithm under Scenario A, representing the ideal case where operational parameters are optimally tuned *a priori*. For this test, the GA was configured with a population size of 200 and a crossover fraction of 0.8, parameters known to be effective for this type of problem. The algorithm was tasked with minimizing the 12-variable Rastrigin function over 200 generations.

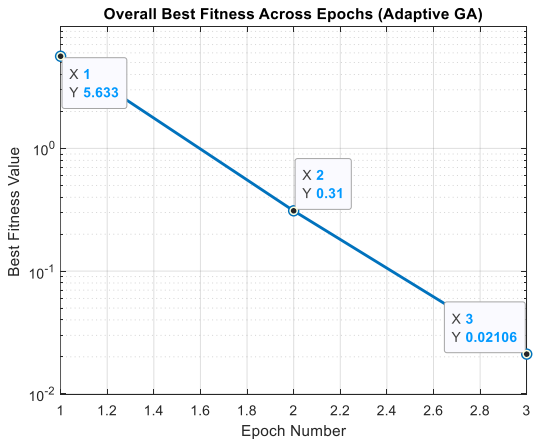

Fig. 4. Step-wise convergence of the adaptive GA across three epochs. The plot highlights the progressive improvement in the best fitness value as the algorithm automatically adjusts its parameters between each 70-generation optimization cycle (Scenario A).

To assess the efficacy of the proposed adaptive genetic algorithm (AGA), it was first evaluated under the same ideal conditions of Scenario A, with identical initial parameters as the static GA (Population Size = 200, Crossover Fraction = 0.8). The AGA was structured to run over three consecutive epochs of 70 generations each, ensuring a fair comparison with a total of 210 generations.

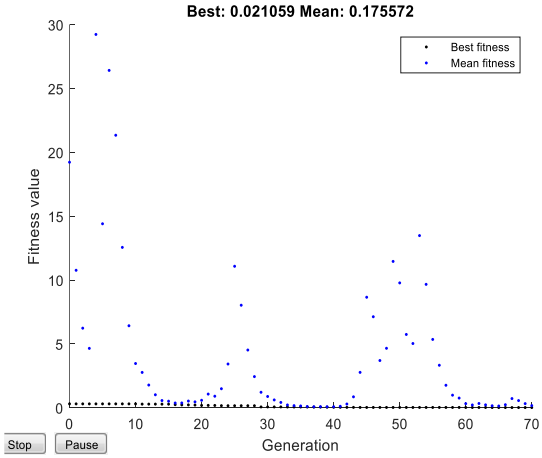

Fig. 5. Final population fitness scores during the third and final epoch of the AGA run. The plot visualizes the convergence behavior over the last 70 generations, confirming the achievement of the final best cost value of 0.02106 (Scenario A).

The results, as summarized in Table 7 and visualized in Figs. 9 and 10, unequivocally demonstrate the superiority of the adaptive approach. The analysis shows that the Adaptive Genetic Algorithm (AGA) achieved a final cost function value of 0.02106 in approximately 8 seconds. It is noted that this represents a minor 2-second increase in computation time, which is attributed to the meta-controller's overhead. However, this marginal cost is far outweighed by the significant enhancement in the quality of the final solution.

A direct comparison conducted in the study reveals that while the optimally-tuned static GA converged to a cost of 0.03, the adaptive framework drove the solution to 0.02106. This constitutes a remarkable improvement of nearly 30%. This powerful outcome serves to confirm two critical points. First, the investigation establishes that the adaptive mechanism does not impose a prohibitive performance penalty. Second, it validates that the algorithm's ability to dynamically refine its search parameters in real-time allows it to uncover superior solutions, even when initialized with settings that were considered favorable for the static version. This finding validates the core hypothesis that intelligent, dynamic adaptation is inherently more powerful than a static configuration.

Having established the efficacy of the adaptive framework under these conditions, the research then proceeds to test the framework's robustness in a more challenging scenario.

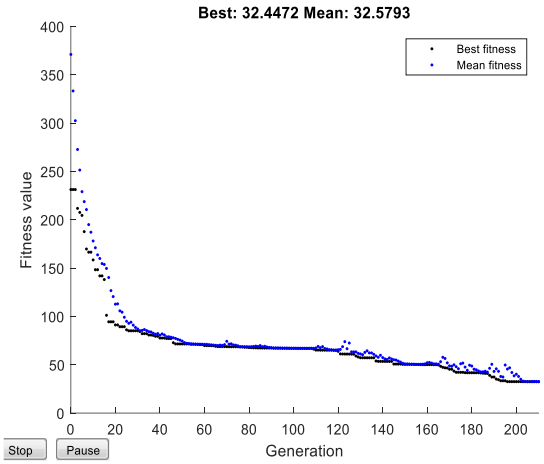

Fig. 6. Genetic optimization algorithm efforts with fixed operators and a fixed initial population of 90 and a fixed crossover rate of 0.9 (Scenario B).

The static GA, constrained by its fixed parameters, failed to overcome the challenge. As depicted in Fig. 11, the algorithm converged prematurely to a significant local minimum. After an

initial drop, the search process stagnated for nearly 180 generations, yielding a poor final fitness value of 32.4472. This result highlights the critical vulnerability of a standard GA to its initial parameter tuning.

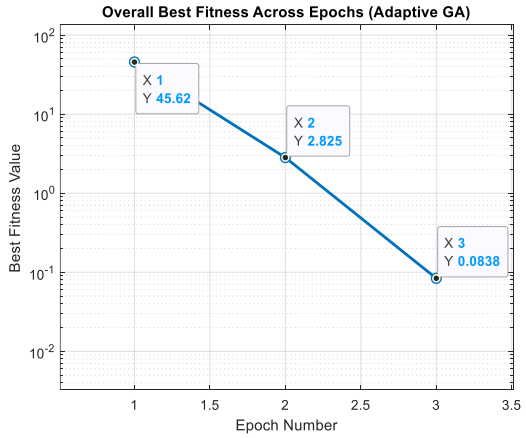

Fig. 7. Step-wise convergence of the adaptive GA across three epochs. The plot highlights the progressive improvement in the best fitness value as the algorithm automatically adjusts its parameters between each 70-generation optimization cycle (Scenario B).

Figure 12 illustrates the self-correcting convergence trajectory of the AGA when initiated with suboptimal parameters. The plot demonstrates the algorithm's ability to escape a local minimum, a feat achieved by dynamically adjusting its search parameters between epochs. This adaptive behavior resulted in a dramatic improvement in the objective function, spanning over three orders of magnitude.

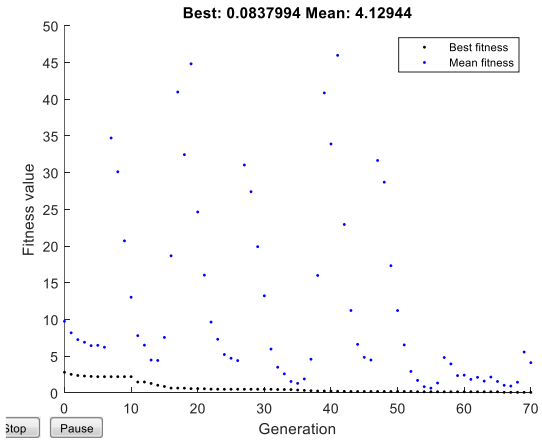

Fig. 8. Final population fitness scores during the third and final epoch of the AGA run. The plot visualizes the convergence behavior over the last 70 generations, confirming the achievement of the final best cost value of 0.08379 (Scenario B).

To complement this, Fig. 13 presents the final population distribution from the last successful epoch, visually confirming the algorithm's strong convergence toward a superior solution region.

Table 2. Comparison of static genetic algorithm (GA) and adaptive genetic algorithm (AGA).

	GA	AGA
F (Scenario A)	0.03	0.021
F (Scenario B)	32.44	0.083
Time (Scenario A)	6sec	8sec
Time (Scenario B)	7sec	9sec

The results demonstrate a significant performance gap when comparing the static GA against its adaptive counterpart, the AGA.

3. SIMULATION

Initially, a precise definition of the problem will be established. This problem entails the selection of optimal values for relevant optimization parameters, including Population Size and Crossover Fraction. Each satellite within the constellation is characterized by six design variables, with two of these variables—namely, the orbital altitude and eccentricity—being held constant. Consequently, a high-dimensional optimization problem emerges, involving a total of forty optimization variables. Subsequently, with each iterative solution attempt, the appropriate parameters for the Genetic Algorithm (GA) optimization technique will be determined. This iterative approach aims to fine-tune the GA parameters to achieve optimal convergence and solution quality.

Table 3. Optimizer parameters under consideration for initial Population Size of 40.

Initial values	Parameter
40	Population Size
0.6	Crossover Fraction

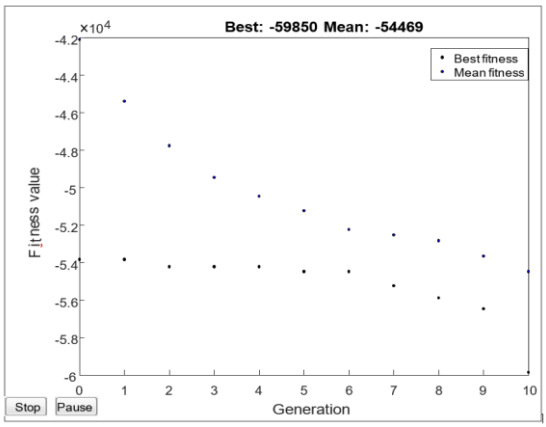

Fig. 9. Performance of the optimization algorithm. for initial Population Size of 40. The simulation took 5 hours.

In the next step, the initial population size will increase in order to optimize the objective function.

Table 2. Optimizer parameters under consideration for initial Population Size of 80.

Initial values	Parameter
80	Population Size
0.6	Crossover Fraction

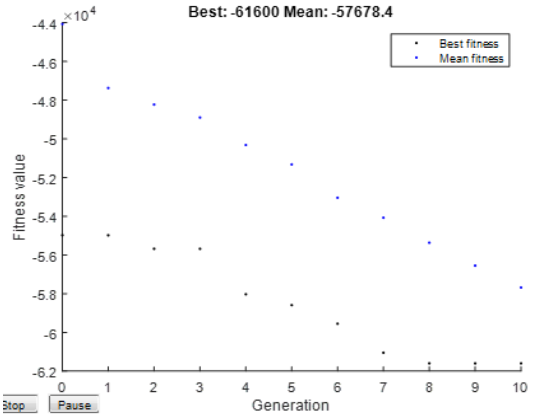

Fig. 10. Performance of the optimization algorithm in optimizing the objective function for initial Population Size of 80.

Performance improvement will be achieved by increasing the population. However, the simulation time took ten hours.

Table 4. Optimizer parameters under consideration for initial Population Size of 100.

Initial values	Parameter
100	Population Size
0.6	Crossover Fraction

The question that arises is: what will be the change in the objective function if, in the next step, the ratio of population changes to the variable is half of the program variables? The results show that despite the increase in optimization time, its performance does not change significantly. (Simulation duration 20 hours).

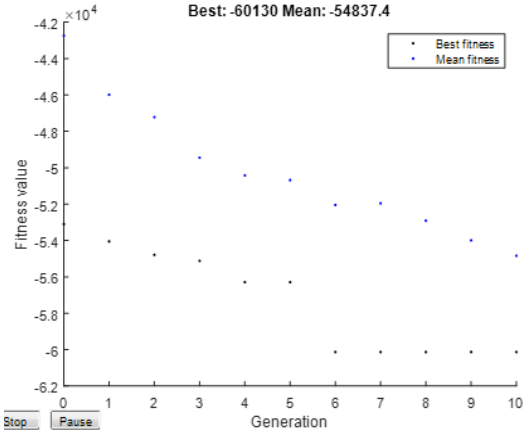

Fig. 11. Performance of the optimization algorithm in optimizing the objective function for initial Population Size of 100.

In the final step, the appropriate initial population size will be examined to obtain a linear relationship between the number of variables and the population size.

Table 5. Optimizer parameters under consideration for initial Population Size of 140.

Initial values	Parameter
140	Population Size
0.6	Crossover Fraction

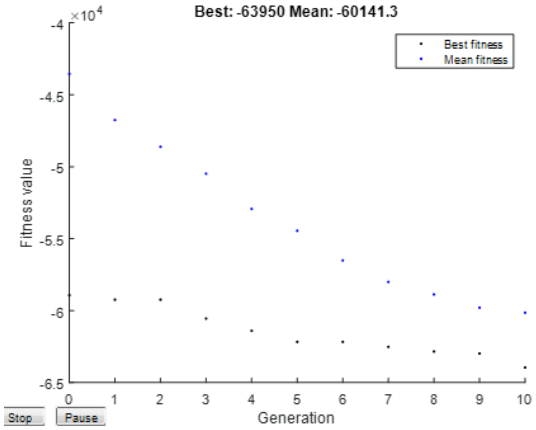

Fig. 12. Performance of the optimization algorithm in optimizing the objective function for initial Population Size of 140.

The performance of the optimization algorithm has shown good performance when the initial population size is 3.5 times the number of main variables. (Simulation duration: 22 hours). In the final step, the effect of CrossoverFraction is examined.

Table 6. Optimizer parameters under consideration for initial Population Size of 140 and Crossover Fraction of 0.5.

Initial values	Parameter
140	PopulationSize
0.5	CrossoverFraction

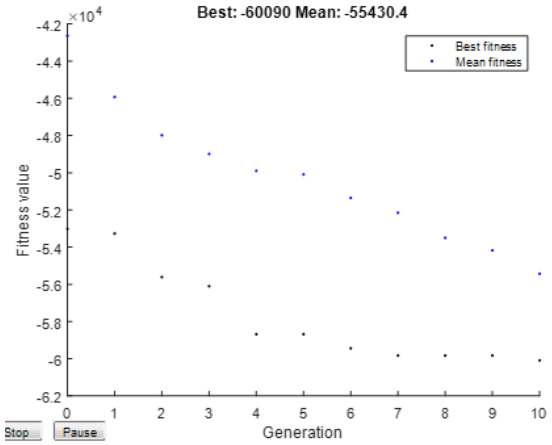

Fig. 13. Performance of the optimization algorithm in optimizing the objective function for initial Population Size of 140 and Crossover Fraction of 0.5.

By changing Crossover Fraction, the performance of the optimization algorithm becomes weak and does not give the desired result.

Table 7. Summary table for Gas.

Number simulation	Population Size	Crossover Fraction	Simulation Time(h)	Objective function	Coverage
1	40	0.6	5	59850s	%69
2	80	0.6	10	61600s	%71
3	100	0.6	20	60130s	%69.59
4	140	0.6	22	63950	%74
5	140	0.5	21	60090	%69.54

Experimental Results and Analysis: A series of simulations was conducted to examine the effects of GA parameters:

- Population Size: Increasing from 40 to 140 improved coverage but significantly extended computation time.
- Crossover Rate: A decrease from 0.6 to 0.5 led to poorer performance, suggesting that maintaining genetic diversity is crucial.
- Computation Time: Larger populations increased optimization time disproportionately, requiring further analysis.

Based on the five simulation scenarios previously analyzed, it is evident that the performance of the static Genetic Algorithm (GA) is highly sensitive to the initial settings of its key parameters, namely the population size and crossover fraction. To address this limitation, this section introduces and evaluates an Adaptive Genetic Algorithm (AGA) framework, where these parameters are dynamically adjusted during the optimization process. To ensure a fair and rigorous comparison, a challenging test scenario was designed. Both the static GA and the AGA are executed for the same number of generations (MaxGenerations=10). Crucially, the AGA is initialized with the parameter values (population size and crossover fraction) that yielded the worst performance in the static GA experiments. This setup is intentionally designed to test the AGA's ability to intelligently navigate and recover from suboptimal initial conditions.

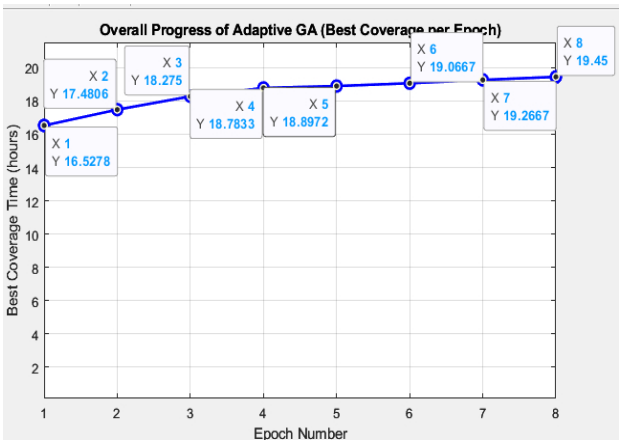

Fig 14. Convergence plot of the adaptive genetic algorithm (AGA) over 8 epochs. Each epoch consists of 10 generations, and the plot displays the best coverage time achieved at the end of each epoch.

Figure 14 clearly demonstrates the efficacy and robustness of the Adaptive Genetic Algorithm (AGA) framework. As illustrated, the AGA, which was initialized with suboptimal parameters corresponding to the worst-case static GA scenario, exhibits a remarkable self-correcting capability. After just two epochs, the algorithm not only recovered from its poor starting conditions but also surpassed the best result achieved by the static GA across all five scenarios, reaching a coverage time of 17.48 hours. This strong convergence trajectory continued, crossing the 18-hour threshold by the third epoch (18.275 hours) and ultimately converging to a highly optimized solution of 19.45 hours after eight epochs. This behavior, characterized by rapid initial improvements followed by fine-tuning in later stages, validates the inherent superiority of a dynamic parameter-tuning mechanism over a static configuration.

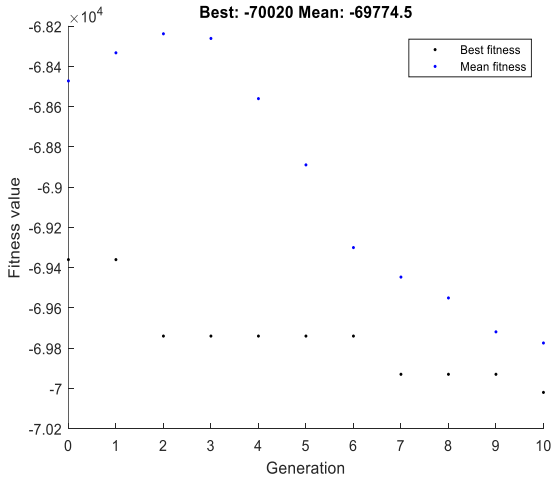

Fig. 15. Detailed optimization process within the final and most successful epoch (Epoch 8). The plot illustrates the evolution of the best solution (Best Fitness), which rapidly converges to the final value of 70020(sec), equivalent to 19.45 hours of coverage.

Figure 14 illustrates the optimization trend at a macro-level, showing the epoch-by-epoch improvement. Complementing this, Fig. 15 provides a micro-level view, detailing the convergence process within the final epoch. Together, these plots offer a comprehensive perspective on the algorithm's performance, demonstrating both its overall progression and the stable convergence within each stage, thus validating the effectiveness of the adaptive approach.

Table 8. Performance comparison of adaptive genetic algorithm (AGA) versus static genetic algorithm (Static GA).

	GA	AGA
Best coverage	17(hour)	19.45(hour)
Total simulation time (hours)	78	15

To evaluate the efficacy of the proposed Adaptive Genetic Algorithm (AGA), its performance was benchmarked against a Static Genetic Algorithm (Static GA) configured with fixed, pre-tuned parameters. The results, summarized in Table 10, demonstrate the unequivocal superiority of the adaptive approach across all evaluated metrics.

Not only did the AGA achieve a higher-quality solution—improving the maximum coverage time by 14.4% from 17.00 to 19.45 hours but it accomplished this in a fraction of the computational time (15 hours vs.78hours). The most telling metric is computational efficiency, which reveals that the AGA was 5.6 times more effective at optimizing the objective function per hour of processing. This remarkable efficiency stems from the algorithm's ability to self-tune its internal parameters, allowing it to avoid redundant exploration of the solution space and intelligently focus computational resources on promising regions. These findings clearly validate the high operational value of AGA for solving complex and time-intensive constellation design problems.

Table 9. Optimized orbital elements for the 10 satellite constellation. Determined by the Adaptive Genetic Algorithm (AGA) to achieve the maximum coverage time of 19.45 hours.

Sat	H(km)	ω	e	I(deg)	RA(deg)	Θ(deg)
1	1200	0	0	53.29	24	79
2	1200	0	0	57.99	178	234
3	1200	0	0	53.29	0.74	307
4	1200	0	0	56.72	150	275
5	1200	0	0	53.86	10	187
6	1200	0	0	52.14	133	155
7	1200	0	0	49.85	63	205
8	1200	0	0	64.74	173	155
9	1200	0	0	52.14	168	358
10	1200	0	0	50.42	17	81

The optimization algorithm intelligently places all satellites in a focused but diverse orbital inclination band (between 49.8 and 64.7 degrees) to maximize temporal coverage over the mid-latitude target region (Tehran) by creating complementary ground paths. Analysis of the optimal values for the RAAN determined by the algorithm reveals a very clever and counterintuitive design strategy for the spatial distribution of orbital planes. Unlike classical systems such as Walker patterns that rely on a symmetrical and uniform distribution of orbital planes over 360 degrees, the optimization algorithm completely departs from this traditional approach. The results obtained (0.74°, 10.31°, 17.76°, 24.06°, 63.60°, 133.52°, 150.69°, 168.46°, 173.05°, 178.76°) indicate a completely asymmetric and irregular distribution that covers the entire orbital space around the Earth. The key point in this strategy is the clustering of orbital planes. Instead of evenly spaced, the algorithm has concentrated the satellites in two main clusters: one cluster near the 0° RAAN and the other cluster near the 180° RAAN, with one satellite acting as a bridge between the two clusters. This arrangement shows that the algorithm has discovered that constant, uniform coverage of all longitudes is neither necessary nor optimal to maximize temporal coverage over a specific target area (Tehran). This cluster structure implements a "waves of coverage" strategy. When a cluster of satellites with closely spaced orbital planes passes over the target area, a long, dense period of coverage is created. Before this wave of coverage ends, the next cluster enters the ground station's field of view from the other side, minimizing the time gaps. This approach demonstrates the true power of evolutionary optimization algorithms, which are able to not only adjust parameter values but also invent entirely new design strategies and provide solutions beyond conventional human design frameworks. In complementing this spatial strategy, the algorithm implemented precise timing through a random and sparse distribution of true anomalies. This ensures that the satellites within each cluster arrive at the target region sequentially and asynchronously (rather than as a single bunch). This asymmetric phasing avoids useless overlap and minimizes time gaps, effectively creating a continuous coverage wave.

4. CONCLUSION

This study addressed the complex, high-dimensional challenge of designing a non-homogeneous LEO telecommunication satellite constellation, formulated as a 30-variable optimization problem aimed at maximizing continuous coverage over a specific region. Recognizing the critical limitations of conventional Static Genetic Algorithms (Static GAs), namely their sensitivity to pre-defined parameters and susceptibility to premature convergence, we introduced a novel Adaptive Genetic Algorithm (AGA). The core innovation of this AGA lies in its dynamic, self-tuning mechanism for critical operator parameters, including population size and crossover rate, which adapt in response to the algorithm's real-time convergence state. To validate the robustness and correctness of this adaptive framework, its performance was first benchmarked against the notoriously difficult, multi-modal Rastrigin test function, where it demonstrated superior exploration and convergence capabilities. Following its validation, the AGA was deployed to solve the constellation design problem. For a rigorous comparison, five distinct scenarios were first executed using a Static GA, which yielded a maximum coverage of 17.00 hours after a prohibitive 78 hours of computation. Subsequently, the AGA was applied to the same problem. The results were decisively superior: the AGA not only achieved a significantly higher coverage of 19.45 hours—a 14.4% improvement—but did so in just 15 hours, representing an 80.8% reduction in computational cost. This translates to a 5.6-fold increase in computational efficiency, unequivocally demonstrating the algorithm's ability to intelligently allocate resources and accelerate convergence towards higher-quality solutions. The findings of this research extend beyond mere numerical improvement. The superior performance of the AGA highlights a fundamental paradigm shift from static, trial-and-error parameter tuning to dynamic, intelligent control in complex engineering optimization. The ability of the AGA to autonomously navigate the vast design space and discover non-intuitive orbital configurations underscores its potential as a powerful tool for next-generation space systems engineering. Future work will focus on integrating multi-objective capabilities into this adaptive framework to simultaneously optimize for coverage, cost, and orbital sustainability, thereby providing a holistic design tool for future indigenous satellite constellation missions.

CONFLICTS OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

[1] Y. Zhang, H. Yang, G. Liang, Y. Chen, Z. Liu, and J. Hu, "Efficient design of constellation for Low-Earth-Orbit object revisit observations," *Aerospace Science and Technology*, vol. 166, 2025, Art. no. 110633, https://doi.org/10.1016/j.ast.2025.110633.

[2] S. Wang *et al.*, "Multi-layer LEO constellation optimization based on D-NSDE algorithm," *Remote Sensing*, vol. 17, no. 6, 2025, Art. no. 994, https://doi.org/10.3390/rs17060994.

[3] C. Qin, Y. Gao, and Y. Wang, "The optimization of low Earth orbit satellite constellation visibility with genetic algorithm for improved navigation potential," *Scientific Reports*, vol. 15, no. 1, 2025, Art. no. 30798, https://doi.org/10.1038/s41598-025-16815-7.

[4] J. Zhang and L. Xing, "An improved genetic algorithm for the integrated satellite imaging and data transmission scheduling problem," *Computers & Operations Research*, vol. 139, 2022, Art. no. 105626, https://doi.org/10.1016/j.cor.2021.105626.

[5] C. Han, S. Bai, S. Zhang, and X. Wang, "Visibility optimization of satellite constellations using a hybrid method," *Acta Astronautica*, vol. 163, Part B, pp. 250-263, 2019, https://doi.org/10.1016/j.actaastro.2019.01.025.

[6] H. Ge, B. Li, L. Nie, M. Ge, and H. Schuh, "LEO constellation optimization for LEO enhanced global navigation satellite system (LeGNSS)," *Advances in Space Research*, vol. 66, no. 3, pp. 520-532, 2020, https://doi.org/10.1016/j.asr.2020.04.031.

[7] A. E. Eiben and J. Smith, *Introduction to Evolutionary Computing*, Latest ed. Berlin, Heidelberg: Springer, 2003, https://doi.org/10.1007/978-3-662-44874-8.

[8] C. Tan, Y. Xu, R. Luo, Y. Li, and C. Yuan, "Low Earth orbit constellation design using a multi-objective genetic algorithm for GNSS reflectometry missions," *Advances in Space Research*, vol. 71, no. 5, pp. 2357–2369, 2023, https://doi.org/10.1016/j.asr.2022.10.035.

[9] Q. Zheng, Y. Cai, and P. Wang, "A modified genetic algorithm for large-scale and joint satellite mission planning," *Egyptian Informatics Journal*, vol. 31, 2025, Art. no. 100713, https://doi.org/10.1016/j.eij.2025.100713.

[10] J. Guo, Y. Wang, X. Xie, and C. Sun, "A fast satellite selection algorithm for positioning in LEO constellation," *Advances in Space Research*, vol. 73, no. 1, pp. 271–285, 2024, https://doi.org/10.1016/j.asr.2023.10.031.