MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 1426-1429

Optimization of Initial Population Size for Enhancing Solution Quality and Reducing Premature Convergence in Genetic Algorithms

Nisha Madaan[1*]
[1]Senior Software Engineer, Department of IT, Tech Mahindra, Greater Noida, Uttar Pradesh, India
*Corresponding Email Id: nishamadaan90@gmail.com
Nisha Nehra[2]
[2]PGT Computer science, Department of Education, Government Model Sanskriti Senior Secondary School Camp, Yamuna Nagar, Haryana, India
Email Id: nisha89nehra@gmail.com

ABSTRACT
Genetic Algorithms (GAs) are robust optimisation techniques derived from natural evolution; nonetheless, their efficacy is contingent upon parameter configuration, particularly population size. An inadequately chosen population size may elevate computing demands or lead to early convergence resulting from diminished genetic diversity. This study optimises the initial population size to enhance solution quality and minimise premature convergence and convergence duration. This research analyses the impact of initial population sizes on genetic algorithm performance and suggests an optimised initialisation strategy that maintains diversity without excessive population growth. A quantitative experiment compares two genetic algorithm variants: BASIC_GA (standard random population initialisation) and NEW_GA (diversity-controlled initial population generated by a minimum distance parameter (δ). A systematic adjustment of population sizes is used to study convergence behaviour and solution quality. Diversity-preserving initialisation in NEW_GA precludes superfluous or clustered individuals. Standard deviation, average fitness, and best fitness value measure algorithm performance across generations. This comparison shows that diversity-aware population initialisation improves genetic algorithm performance and reduces premature convergence. Greater populations do not inherently yield superior solutions, and meticulously optimising population size can enhance genetic algorithm performance. This study suggests that the starting population size is a critical element in genetic algorithms that influences convergence and solution quality. Diversity-controlled initialization improves genetic algorithm convergence and robustness without added computational cost.
Keywords: *Genetic Algorithm Parameter, Initial Population, Optimization, Population Size, Convergence Time.*

INTRODUCTION
Genetic algorithms (GAs) are population-based optimisation methods inspired by natural evolution, capable of efficiently exploring large, nonlinear, and multimodal search spaces without gradient information (Taha et al., 2025; Vie et al., 2020). Their population structure enables parallel exploration, making GAs widely applicable in areas such as feature selection, cloud computing, industrial optimisation, and healthcare analytics (Fofanah et al., 2023). However, GA performance is strongly influenced by algorithmic parameters, particularly initial population size and diversity, which affect convergence speed, computational cost, and solution quality. Low diversity can cause premature convergence to local optima, while excessively large populations increase computational burden with limited performance gains (Vie et al., 2020).
Recent studies highlight that the quality and structure of the initial population shape the evolutionary trajectory by influencing selection, crossover, and mutation processes. Diversity-enhancing and informed initialization strategies, such as regression-based approaches, have been shown to improve convergence and avoid redundant searches (Hassanat et al., 2018; Seghir & Khababa, 2018). Despite extensive research on genetic operators and hybrid methods, population sizing remains underexplored, even though it critically governs the balance between exploration and exploitation and GA robustness relative to other evolutionary methods (Taha et al., 2025; Fofanah et al., 2023). This study therefore focuses on optimizing initial population size to enhance solution quality and prevent premature convergence.

LITERATURE REVIEW
Díaz-Álvarez et al. (2022) showed that population size in evolutionary algorithms has a non-linear impact on energy consumption, with larger populations increasing cache misses and power use without proportional gains in solution quality. Their study highlights energy efficiency as a critical optimisation objective alongside time and accuracy, concluding that careful parameter tuning is essential for efficient evolutionary computation.
Similarly, **Langazane and Saha (2022)** demonstrated that optimisation performance in power system protection is highly sensitive to algorithm parameters. Their results indicate that lower population sizes, along with modest crossover (30%) and mutation (2%) rates, significantly improve genetic algorithm convergence and fitness, while swarm size and inertia weight dominate particle swarm optimiser performance. Comparing particle swarm optimisation to genetic algorithms shows that the former parameter setting outperforms the latter. Particle swarm optimisation reduces relay operational speed by 15% and maximises system selectivity. The particle swarm optimisation algorithm's capacity to avoid premature convergence, consistency, and efficiency were demonstrated by its optimal protection coordination.
Zheng et al. (2022) stated that the non-dominated sorting genetic algorithm II (NSGA-II) is the most often utilised MOEA in real-world applications. Unlike several simple MOEAs analysed mathematically, the NSGA-II has not been studied. We demonstrate that NSGA-II mathematical runtime analyses are possible in this work. We show that the NSGA-II with two classic mutation operators and three ways to select the parents meets the same asymptotic runtime guarantees as the SEMO and GSEMO algorithms on the basic OneMinMax and LOTZ benchmark functions with a population size a constant factor larger than the Pareto front size. If the population size is only equal to the Pareto front, the NSGA-II cannot efficiently compute the complete front (for an exponential number of iterations, the population will always miss a constant proportion). Our experiments corroborate these findings.
Lange et al. (2023) described genetic algorithms as black-box optimisation algorithms based on biological evolution. While conventional genetic algorithms are general-purpose and biologically inspired heuristics, recent work enables data-driven discovery of novel GAs through flexible parameterisation of genetic operators. By modelling selection and mutation-rate adaptation with attention mechanisms and optimising them via meta–black-box optimisation, learned genetic algorithms outperform adaptive baselines and generalise across unseen problems, dimensions, and budgets. Ablation studies confirm that modular, learnable operators can be effectively integrated into standard evolutionary frameworks.
In contrast, **Wietheger and Doerr (2024)** report that NSGA-II remains the most widely used multi-objective evolutionary algorithm and performs well for bi-objective problems, but its effectiveness declines as the number of objectives increases. A mathematical examination revealed that the NGSA-II misses a constant factor of the Pareto front in the m-objective OneMinMax issue when $m > 3$. This study presents the first mathematical runtime analysis of the NSGA-III, a revision of the NSGA-II that handles more than two objectives better. We demonstrate that the NSGA-III with enough reference points, a modest constant factor larger than the Pareto front, computes the complete Pareto front of the 3-objective OneMinMax benchmark in $O(n \log n)$ iterations. This holds for all population sizes (at least Pareto front size). This benchmark gives the NSGA-III a huge edge over the II.
Sécheresse et al. (2025) found that Large Language Models (LLMs) perform well across tasks but depend on input prompts. Manual prompt engineering is effective but inefficient. GAAPO introduces a hybrid, evolutionary framework that automatically optimises prompts using genetic algorithm principles and multiple prompt-generation strategies beyond simple mutation and crossover. Experiments on ETHOS,

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 1426-1429

MMLU-Pro, and GPQA show key trade-offs between population size and generations, the influence of selection stability, and the role of model capacity in generating transferable prompts. Results demonstrate improved prompt optimisation efficiency and enhanced LLM performance, advancing automated prompt engineering methods.

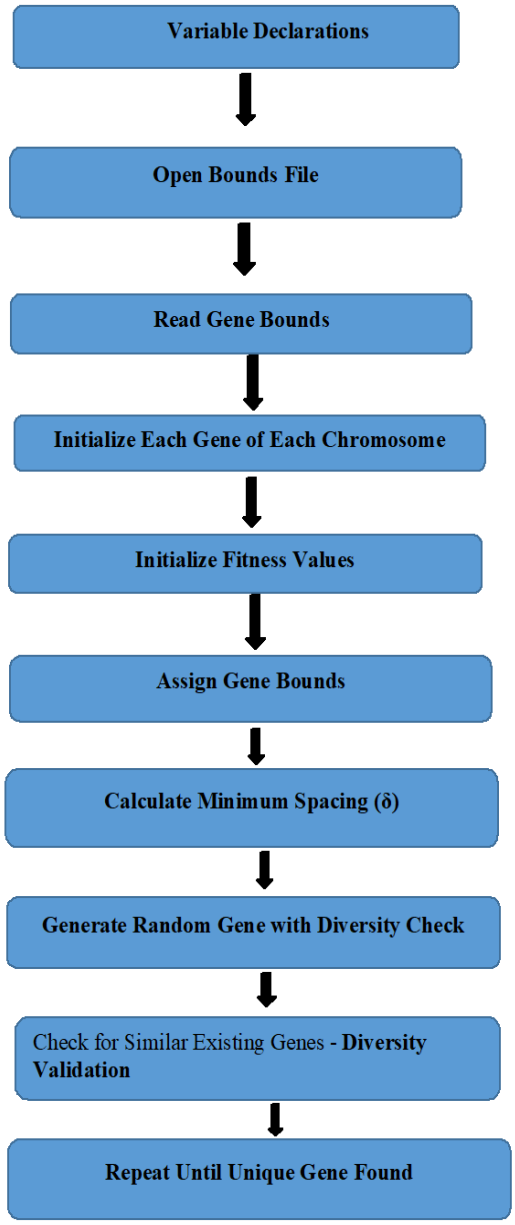

Figure 1. Proposed Scenario.

Research Gap

Despite extensive research on genetic operators, hybridisation, and adaptive mechanisms, diversity-regulated initial population generation remains underexplored in genetic algorithm studies. Most work assumes random initialisation, overlooking its influence on premature convergence and solution quality. Few studies systematically compare standard and diversity-aware genetic algorithms under equivalent conditions, leaving the generational impact of diversity-preserving initialisation insufficiently examined.

METHODOLOGY

This quantitative experiment compares BASIC_GA and NEW_GA genetic algorithm versions under identical settings. The BASIC_GA generates the initial population using randomisation, while the NEW_GA utilises a minimum spacing parameter (δ) to regulate diversity. Keeping all genetic operators and parameters the same except for initialisation allows for a fair and controlled comparison. The NEW_GA maintains a minimal gene value difference between individuals during initialisation to ensure population variation. Calculate δ spacing parameter based on maximal gene value and population size. Avoid repetition by setting the minimal gene value difference. Population generation allows gene values with an absolute change from existing values greater than or equal to δ. Otherwise, the gene is discarded and regenerated. This method equally distributes individuals across the search space and eliminates early evolutionary grouping of similar solutions.

BASIC_GA and NEW_GA evolve similarly after population initialisation. After determining fitness for everyone, parent chromosomes are chosen. Standard crossover and mutation activities make offspring and update the population iteratively. Evolution continues until the maximum number of generations.

All experiments are computationally controlled for consistency. Between experiments, crossover probability, mutation probability, number of variables, and maximum generations are fixed, but population size is modified to examine algorithm performance. BASIC_GA and NEW_GA are compared by best fitness, average fitness, and standard deviation across generations. Average fitness measures population performance, best fitness ideal solution quality, and standard deviation population variation and convergence stability. The convergence behaviour and diversity-preserving initialisation are fully described by these measures. This study compares BASIC_GA and NEW_GA for multiple population sizes. This approach highlights the importance of δ-based spacing in improving exploration, solution quality, and decreasing premature convergence by separating the impact of diversity-controlled initialisation.

Figure 1 shows a proposed scenario for diversity-preserving Genetic Algorithm initialisation employing a minimum spacing parameter (δ) to produce a separate population and prevent premature convergence. After creating a valid population, GA operations—selection, mating, crossover, and mutation—are repeated until the termination requirement is met.

The step by step implementation is explained below:

Step 1: Variable Declarations: Population size (POPSIZE), number of variables (NVARS), maximum generations (MAXGENS), chromosome and gene arrays, fitness values (raw, relative, and cumulative), and distance and diversity auxiliary variables must be declared.

Step 2: Open Bounds File: Open the gene decision variable lower and upper boundaries input file. The optimisation problem's feasible search space is defined by this file.

Step 3: Read Gene Bounds: Read and store the lower bound (lbound[i]) and upper bound (ubound[i]) for each gene i to constrain all variables inside their ranges.

Step 4: Initialize Each Gene of Each Chromosome: Use a pseudo-random number generator within the constraints to generate an initial gene value for each population chromosome and gene.

Step 5: Initialize Fitness Values: Set each chromosome's raw, relative, and cumulative fitness to zero. This ensures that fitness values are examined once the initial population is formed.

Step 6: Assign Gene Bounds: To ensure feasibility during initialisation and genetic operations, assign lower and upper boundaries to each gene.

Step 7: Calculate Minimum Spacing (δ): Use the relation to calculate the minimum spacing (δ) between gene values to maintain diversity:

$$\delta = \frac{\text{ubound[i]}}{\text{POPSIZE}}$$

where POPSIZE is population size and ubound[i] is gene upper bound. The δ parameter determines the minimal gene value difference between population members. A well-structured gene value population is guaranteed by this requirement. To maintain population diversity, gene values whose difference from existing values is smaller than δ are rejected and regenerated during initialisation.

Adding δ-based spacing during population initialisation enhances genetic algorithm performance. It preserves original population variety, inhibits solution clustering, and expands search space. Early generation redundancy reduction reduces premature convergence and improves convergence stability and solution quality.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 1426-1429

Step 8: Generate Random Gene with Diversity Check: Create a random gene value inside the boundaries and apply a diversity constraint using the derived δ value.
Step 9: Check for Similar Existing Genes
Step 9.1: Diversity Validation: Reject the gene and regenerate a new value if the absolute difference between the new and old gene values is smaller than δ. Otherwise, accept the gene value and continue.
Step 10: Repeat Until Unique Gene Found: Repeat 8 and 9 until a gene value meets the diversity constraint. Continue until all genes of all chromosomes are initialised, creating a diversified and well-distributed initial population.

RESULTS AND DISCUSSIONS

By adjusting the starting population size to 20, 30, and 40 while maintaining other parameters (MAXGENS = 5, NVARS = 3, PXOVER = 0.8, PMUTATION = 0.15), the diversity-aware New Genetic Algorithm (NEW_GA) was compared to BASIC_GA. Performance metrics included best fitness value, average fitness, standard deviation, and premature convergence resistance. The results show that population size affects solution quality and convergence.

Case 1: Population Size = 20

With 20 individuals, the NEW_GA had a best fitness value of 1704, whereas the BASIC_GA had 277.64. NEW_GA's higher best fitness suggests better search space exploration due to the delta-based diversity constraint during initialisation. NEW_GA had stronger fitness stability over generations, while BASIC_GA stagnated early, indicating premature convergence. Table 1 shows this in more detail.

Table 1. Population Size = 20.

Algorithm	Best Fitness – Average Fitness	Std. Deviation
NEW_GA	1704	382.96
BASIC_GA	277.64	382.96

Case 2: Population Size = 30

When the population reached 30, the performance gap grew. The best fitness was 3068 for NEW_GA and 820.94 for BASIC_GA. Although population size increased genetic diversity, BASIC_GA still had redundant individuals owing to random initialisation. NEW_GA used minimum spacing (delta) to manage individual similarity, resulting in persistent improvement across generations.

Table 2. Population Size = 30.

Algorithm	Best Fitness – Average Fitness	Std. Deviation
NEW_GA	3068	955.12
BASIC_GA	820.94	100.47

Case 3: Population Size = 40

At 40 individuals, NEW_GA had the highest best fitness value of 4391, whereas BASIC_GA had 1559.71. NEW_GA rejected closely spaced individuals during initialisation to efficiently use population size, despite higher computing cost. This increased variety without redundancy, minimising premature convergence.

Table 3. Population Size = 40.

Algorithm	Best Fitness – Average Fitness	Std. Deviation
NEW_GA	4391	1143.04
BASIC_GA	1559.71	270.07

The NEW_GA has better convergence reliability and fitness than the BASIC_GA across all population sizes. Optimising the initial population through controlled diversity improves solution quality, scalability, and convergence stability, while reducing premature convergence. Compared with the basic GA, the proposed GA achieves higher fitness with better efficiency by accounting for population size, eliminating redundancy, and maintaining genetic variability. This prevents population homogeneity, enabling continued exploration and the generation of superior solutions. In Figure 2, this study observe that the optimal fitness value varies with population size for both New GA and Basic GA. This variance is shown in proportion to the population size. Figure 2 shows that the New Genetic Algorithm improves solution quality by steadily and significantly increasing best fitness values from 20 to 40. Diversity-aware initialisation improves exploration and prevents premature convergence, as seen by this trend.

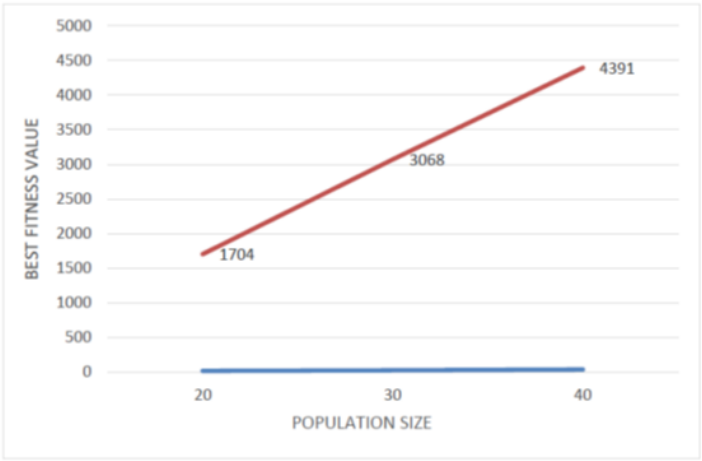

Figure 2. New Genetic Algorithm.

Figure 3 indicates that the New Genetic Algorithm's best fitness values grow from 20 to 40, indicating improved search capabilities with bigger populations. However, the lower fitness values indicate limited variety utilisation, demonstrating how population size affects convergence efficiency.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 1426-1429

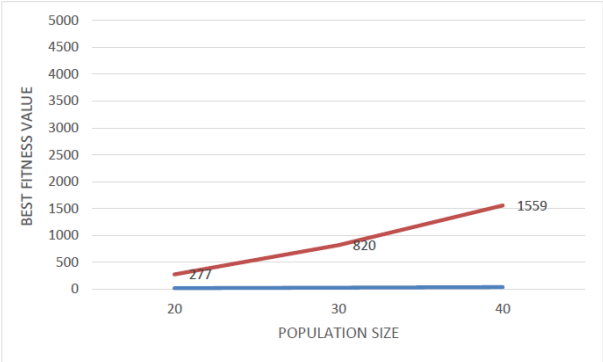

Figure 3. Basic Genetic Algorithm.

This study used diversity-aware initialisation to change population size to improve solution quality and avoid premature convergence, supporting current literature. Al-Terkawi and Migliavacca (2025) dynamically controlled GA parameters in large-scale distributed systems using an automated parallel genetic algorithm to improve convergence efficiency. Population size optimisation enhances convergence and fitness before adaptive approaches, enhancing parallelism and adaptive control research. Both studies reveal population-related variables drive GA durability and scalability. Population size and encoding probabilities affect GA performance and waste computer resources, according to O'Dwyer and O'Brien (2025). Figures 2 and 3 show that larger, well-structured populations had higher best fitness values than smaller or poorly initialised populations. Diversity-preserving initialisation improves efficiency without changing encoding methods, unlike probabilistic.
Braghin et al. (2025) found that GA efficacy in restricted multi-objective optimisation requires diversity to explore competing objectives. Figure 2's better performance confirms their findings that optimum population size and reduced spacing avoid premature convergence by managing variety. Recent work prioritizes adaptability and multi-objective optimisation, but evidence shows that optimizing initial population size is a simple, effective way to improve GA convergence and solution quality.

CONCLUSION

This study shows that diversity-aware initialisation improves GA solution quality and prevents premature convergence across population sizes of 20–40. Adjusting early individual distribution enhances search and convergence: small populations underexplore, while large randomly initialised populations suffer redundancy. Preserving early genetic diversity reduces crowding, expands search coverage, and yields higher fitness and more stable convergence as population size increases. These results show that Genetic Algorithm optimisation should include population size as a design parameter rather than a static parameter. This study observed that diversity-preserving initial population size optimisation improves Genetic Algorithm efficiency without increasing computational cost. The suggested approach enables researchers and practitioners create efficient, reliable, and scalable optimisation solutions for complicated, high-dimensional issues. Adding adaptive population sizing, multi-objective, and real-time optimisation could expand this research.

REFERENCES

1. Taha, Z. Y., Abdullah, A. A., & Rashid, T. A. (2025). Optimizing feature selection with genetic algorithms: a review of methods and applications. Knowledge and Information Systems, 1-40.
2. Hassanat, A. B., Prasath, V. S., Abbadi, M. A., Abu-Qdari, S. A., & Faris, H. (2018). An improved genetic algorithm with a new initialization mechanism based on regression techniques. Information, 9(7), 167.
3. Vie, A., Kleinnijenhuis, A. M., & Farmer, D. J. (2020). Qualities, challenges and future of genetic algorithms: a literature review. arXiv preprint arXiv:2011.05277.
4. Seghir, F., & Khababa, A. (2018). A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. Journal of Intelligent Manufacturing, 29(8), 1773-1792.
5. Fofanah, A. J., Koroma, S., & Bangura, H. I. (2023). Experimental exploration of evolutionary algorithms and their applications in complex problems: genetic algorithm and particle swarm optimization algorithm. British Journal of Healthcare and Medical Research-Vol, 10(2).
6. Díaz-Álvarez, J., Castillo, P. A., Fernández de Vega, F., Chávez, F., & Alvarado, J. (2022). Population size influence on the energy consumption of genetic programming. Measurement and Control, 55(1-2), 102-115.
7. Langazane, S. N., & Saha, A. K. (2022). Effects of particle swarm optimization and genetic algorithm control parameters on overcurrent relay selectivity and speed. IEEE Access, 10, 4550-4567.
8. Zheng, W., Liu, Y., & Doerr, B. (2022, June). A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In Proceedings of the AAAI conference on artificial intelligence (Vol. 36, No. 9, pp. 10408-10416).
9. Lange, R., Schaul, T., Chen, Y., Lu, C., Zahavy, T., Dalibard, V., & Flennerhag, S. (2023, July). Discovering attention-based genetic algorithms via meta-black-box optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 929-937).
10. Wietheger, S., & Doerr, B. (2024, July). A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 63-64).
11. Sécheresse, X., & Villedieu de Torcy, A. (2025). GAAPO: genetic algorithmic applied to prompt optimization. Frontiers in Artificial Intelligence, 8, 1613007.
12. Al-Terkawi, L., & Migliavacca, M. (2025). An automated parallel genetic algorithm with parametric adaptation for distributed data analysis. Scientific Reports, 15(1), 10836.
13. O'Dwyer, D. W., & O'Brien, E. J. (2025). Genetic algorithm encoding probabilities & population size. WIT Transactions on Information and Communication Technologies, 20.
14. Braghin, A., Galuppi, L., & Royer-Carfagni, G. (2025). Evaluation of a genetic algorithm for constrained multi-objective structural optimization in laminated glass design. Composite Structures, 354, 118773.