A Scale-free Network-based Genetic Algorithm with Balanced Exploration and Exploitation

Yazid Hoblos*, Charbel Fayad*, Danielle Azar, and Eileen Marie Hanna

Department of Computer Science and Mathematics
Lebanese American University, Byblos, Lebanon

*Equal contribution

Abstract

Genetic algorithms (GA) are widely used to solve complex computational problems. They are inspired by natural evolution, and thus belong to the family of evolutionary algorithms. Although GA shows superior performance in many applications, its traditional panmictic version lacks restrictions among individuals, resulting in a lack of communication both intra- and inter-generationally. This paper introduces a novel network-based approach that utilizes phenotypic and genotypic similarities to establish inter-chromosomal links. Based on the scale-free property of the Barabasi-Albert (BA) model, we dynamically assign authority nodes to drive the evolutionary process. Comparative evaluations against panmictic GA and a previously developed network-based genetic algorithm demonstrate favorable results for our suggested approach. The assessments involve the average best fitness over 50 runs conducted on eight well-known benchmark functions. The implementation is freely available at: `https://github.com/yazid-hoblos/ENGA.git`.

Keywords: Evolutionary algorithms, Network science, Genetic algorithm, Population structure

1 Introduction

Meta-heuristic algorithms are used in many fields to solve complex optimization functions. They are mainly categorized into single-solution-based and population-based algorithms. Of the latter type, evolutionary algorithms are the most studied form [26]. Following its introduction, the genetic algorithm (GA) [25] has attracted much attention and has quickly become widely used. Similar to other evolutionary algorithms, GA design is inspired by biological evolution. It utilizes the crossover and mutation operators to create explorative diversity while repeatedly selecting the fittest individuals to drive the evolutionary process. The process starts with an initial generation that comprises a pool of chromosomes holding sequences of genes, representing the solution for a given problem. These gene sequences are typically randomly generated at the start.At each iteration, a set of chromosomes is selected to create the next generation until reaching certain stopping criteria. The crossover operator creates the new individuals as a recombination of two selected individuals, while

the mutation operator allows for the constant introduction of novel genes to enhance the explorative power of the algorithm [34]. This design has proven to be effective in solving many problems in different contexts, including medicine and engineering [21, 7]. In many cases, this standard GA design has been extensively modified to achieve better results for different applications [43]. In addition to the variations at the encoding level, multiple variations of the selection, crossover, and mutation operators have been proposed [3]. GA was also modified to accommodate for the multi-objectivity required in many contexts [28].

In its standard form, GA lacks communication between the chromosomes both intra- and inter-generationally. We claim that this lack represents a major drawback that limits the capacity of GA to achieve its ideal performance. This being said, the standard GA is not completely devoid of such communication as it is implicit in the selection process. Its design and interpretation would vary between the different GA variations and selection approaches, yet it is generally the case that better chromosomes have a higher probability of mating together and of mating in general. Despite the efficacy of this approach when applied to many problems, it could be significantly enhanced by a more intricate definition of inter-chromosomal relations. This enhancement could be made at three main levels: (1) creating interacting communities of solutions, (2) establishing an inter-generational memory for the network of solutions, (3) and recruiting authority nodes that could drive the evolutionary process in better directions.

The establishment of several communities that interact with each other by means of chromosomal migration has already been used by the Island GA variation [24]. These communities could be defined based on both the phenotypic and genotypic content similarities between the chromosomes. The main advantage of this approach is the maintenance of diversity and the enhanced explorative potential of the algorithm [32]. However, two major limitations of this approach are the need to pre-specify the number of communities to be established and the maintenance of these communities throughout the evolutionary process. This could lead to the obstruction of the evolutionary process by the maintenance of communities that should be eradicated, or the merging of communities that should be separated. Ideally, the model should be able to dynamically establish and evolve its structural communities as needed using both the phenotypic and genotypic similarities. Moreover, in the standard GA and most of its common variations, generations are isolated from each other, and no generational memory is maintained. Alternatively, a short-term memory could be provided for each generation, which would allow it to avoid the mistakes of the previous generations and evolve to explore better regions of the search space.

Furthermore, the population of chromosomes could be mapped to a network architecture which would allow for the emergence of authority nodes in the network representation of the solutions.

The exceptional connectivity of these nodes allows them to play a leading role in the creation of the chromosomes of the following generation. In the case of multiple communities in the network, representative nodes from each community could play the role of authority nodes and drive the re-population process. In standard GA, the most fit chromosomes play this role since they are more likely to be selected for mating. This tendency results in the individuals of the subsequent generations becoming more similar to these fit chromosomes. However, this has the risk of leading to early convergence as the chromosomes will rapidly converge to a narrow range of solutions. To avoid this risk, the authority nodes in the network must be carefully selected so that they can drive

the solutions to better explore the search space and to approach the desired solutions more efficiently once they get in their vicinity. To maintain such a balance of exploration and exploitation, the choice of authority nodes must depend on both the genotypic and phenotypic components of the chromosomes. This is necessary to allow for the maintenance of the diversity necessary for exploration while allowing a subset of the chromosomes to converge to narrow regions around the approached optimal values to guarantee effective exploitation.

In our proposed approach, we use a scale-free network to establish authority nodes in the network. We rely on the Barabasi-Albert (BA) model to create the scale-free topology to which we repeatedly assign the individuals of successive generations. Moreover, we base the assignment of each individual to its corresponding node on a score that combines its phenotypic and genotypic importance in its generation. We define the phenotypic and genotypic importance of an individual as its normalized fitness value and average genetic similarity with all the other individuals, respectively. This score is repeatedly computed for each individual across all generations in order to establish the inter-chromosomal links and recruit the authority nodes for each generation. The phenotype and genotype components of this score create a balance between exploration and exploitation. We further add weights to control the contribution of each of these two components to the scores of the nodes, and we allow these weights to evolve dynamically starting with a higher contribution from the genotypic component and ending with a predominant dependence on the phenotypic one. This dynamic evolution of the weights results in a shift from enhanced exploration at the beginning to increased focus on exploitation as the algorithm progresses. Overall, the inter-chromosomal links established in a scale-free topology allow the nodes with higher phenotypic and genotypic importance to drive the evolutionary process in a way that maintains a balance of exploration and exploitation.

2 Literature Review

In recent decades, several metaheuristic-based algorithms aiming to search for approximate solutions [9] have been proposed for numerous optimization problems [15]. Evolutionary algorithms are one important class of such algorithms. They are inspired by natural evolution [33], and have gathered attention due to their impressive capability in navigating the search space of optimization problems and producing promising results [14]. The genetic algorithm (GA), as introduced by Holland [25], is a widely acknowledged conventional evolutionary algorithm designed to emulate the biological principle of the survival-of-the-fittest. GA has gained recognition for its efficiency in finding approximate solutions to complex problems within a relatively short time frame [23, 19, 41].

In the standard genetic algorithms, the population is typically conceived as a panmictic, unordered set of individuals, where each individual entity has the opportunity to mate with another entity and produce offspring that would inherit genetic traits from their parents [30]. However, it can be argued that this representation might not precisely reflect real-world scenarios [32]. The combination of a finite population size and unrestricted mating may lead to genetic drift, thereby reducing the genetic diversity within the pool. Diversity, in this context, refers to the variations among individuals in the population [4]. Maintaining higher population diversity is crucial to avoid premature convergence allowing the achievement of better results [39, 42, 11]. Furthermore, to maintain diversity, it is crucial to balance exploration and exploitation [6, 45]. Exploration refers to the algorithm's ability to explore new areas in the search space, while exploitation concerns the algorithm's

capacity to search these regions effectively [13, 45]. Essentially, the maintenance of a diverse population reflects the explorative potential of the algorithm, while its momentum in approaching the optimal solution highlights its exploitative efficiency [2, 45].

It is widely acknowledged that in most systems structure and behavior are fundamentally interconnected given the structure's tendency to impose constraints on behavior [8]. Consequently, a considerable body of literature suggests various methods for implementing a population structure that constrains the behavior of genetic algorithms by reducing intra-group diversity in an attempt to increase the overall inter-group diversity [30]. Organizing the chromosomes into structured topologies and selecting offspring from local neighborhoods clearly affect the information flow and selection pressure [40], ultimately enhancing the performance of the algorithm [37, 38, 36, 35]. These topologies facilitate space reorganization through two distinct mechanisms: spatial segregation, which mirrors the geographical separation of sub-populations in the natural world, and spatial distance, which restricts the individual's mating options to the nearby individuals [32]. One example of a spatially structured population in genetic algorithms is the Island-based Genetic Algorithm [24]. In this approach, independent populations (islands) evolve separately, with the ability to exchange chromosomes between each other. The migration of individuals with high fitness levels across sub-populations plays a pivotal role in increasing the selective pressure and contributing to the maintenance of genetic diversity within local sub-populations [10]. This model has proven to be effective in solving complex problems that the standard model struggles with [12]. Another spatially structured population model is the cellular Genetic Algorithm (cGA) [1], in which individuals are organized in a grid that limits their interactions to their local neighborhood. This model enhances population diversity and exploration through the slow diffusion of solutions [6]. Similarly, the terrain-based model, a self-tuning version of cGA, interprets the dimensions of the grid as the mutation rate and the number of crossover points [18]. In other models such as the hierarchical hypercube model, introduced in [6], the population structure can dynamically construct its hierarchy. Some hybrid models that combine several of the presented approaches were also developed. For instance, the authors in [29] proposed the Patchwork model, a combination of the island and cellular models, which allows for varying population sizes, providing better control over the population diversity and selective pressure by relying on local operations and the incorporation of behavioral rules for individuals.

Finally, several graph-based population structure models of genetic algorithms have been proposed [17]. Of these, irregular population structures, such as random graphs, small-world networks, and scale-free networks, have gained attention. Scale-free networks, in particular, tend to capture real-world properties accurately [5]. In [20] the authors applied a GA with a scale-free population structure to the mobile robot localization problem. Similarly, in [27], they attempted to use GA with a scale-free population structure to solve bi-objective optimization problems. In [22], a cGA with a scale-free population structure is proposed. Moreover, a networked genetic algorithm (NGA) that models the population structure of a steady-state genetic algorithm and relies on a heterogeneous network structure with information fusion strategies to restrict mating to neighbors in a scale-free network was introduced in [16].

3 Methodology

In this section, we formally define a general optimization problem. Additionally, we outline the framework of the traditional genetic algorithm and present the Barabasi-Albert (BA) algorithm. Subsequently, we explain our proposed algorithm by providing details about the process of creating the network and evolving the population. Furthermore, we describe how we experimentally assessed the effectiveness of our framework on various benchmark functions.

3.1 Problem formulation

A optimization problem can be expressed as either a maximization or minimization problem of an objective function, where the objective function serves as an exact or approximate (heuristic) measure of the solution's fitness. It can be formally represented as $P = (S, f)$, $X = x_1, x_2, \ldots, x_n$ denoting the set of variables, and $D_1, D_2, \ldots, D_n$ the corresponding domains for the variables. The search space of solutions, S, is defined as:

$$S = \{s = \{(x_1, v_1), (x_2, v_2), \ldots, (x_n, v_n)\} \mid v_i \in D_i\} \tag{1}$$

Given a solution s in the search space, it is considered a candidate solution formed by n variables each with a certain value v that corresponds to that variable domain. The quality of these candidate solutions is evaluated using an objective function f, where $f(s) = y \mid s \in S, y \in \mathbb{R}$. Solving an optimization problem P involves finding an optimal solution $s^* \in S$ such that $f(s^*) \leq f(s)\ \forall s \in S$. Typically, finding a global optimum like s^* is challenging, and obtaining an approximate solution is considered satisfactory. Genetic algorithms, as evolutionary metaheuristics, are particularly adept at effectively addressing such optimization problems.

3.2 Genetic algorithm

In genetic algorithms, a candidate solution s is represented as a chromosome of genes, where each gene corresponds to a variable x_i. Using this representation, in the initialization phase we generate a random population of a fixed size, N, resulting in the first-generation population of chromosomes. The first step involves evaluating the fitness of all the chromosomes using the fitness function f. To create the new population, pairs of chromosomes are selected based on some fitness-driven scheme, such as roulette selection or tournament selection [44]. Following that, the crossover and mutation operators are applied to evolve the selected chromosomes, which are subsequently added to the new population. After the selection phase is complete, we replace the old population with the new one and repeat the initial step until a stopping criterion is met. Typically, the stopping criterion is a maximum number of generations if the genetic algorithm fails to reach the optimal solution s^*. This process is illustrated in Fig. 1 presenting all the steps of the genetic algorithm framework.

3.3 Barabasi-Albert model

Before delving into our proposed algorithm, it is essential to understand how a scale-free network can be created using the Barabasi-Albert algorithm. Barabási and Albert originally developed the BA algorithm by examining how new nodes connect to existing

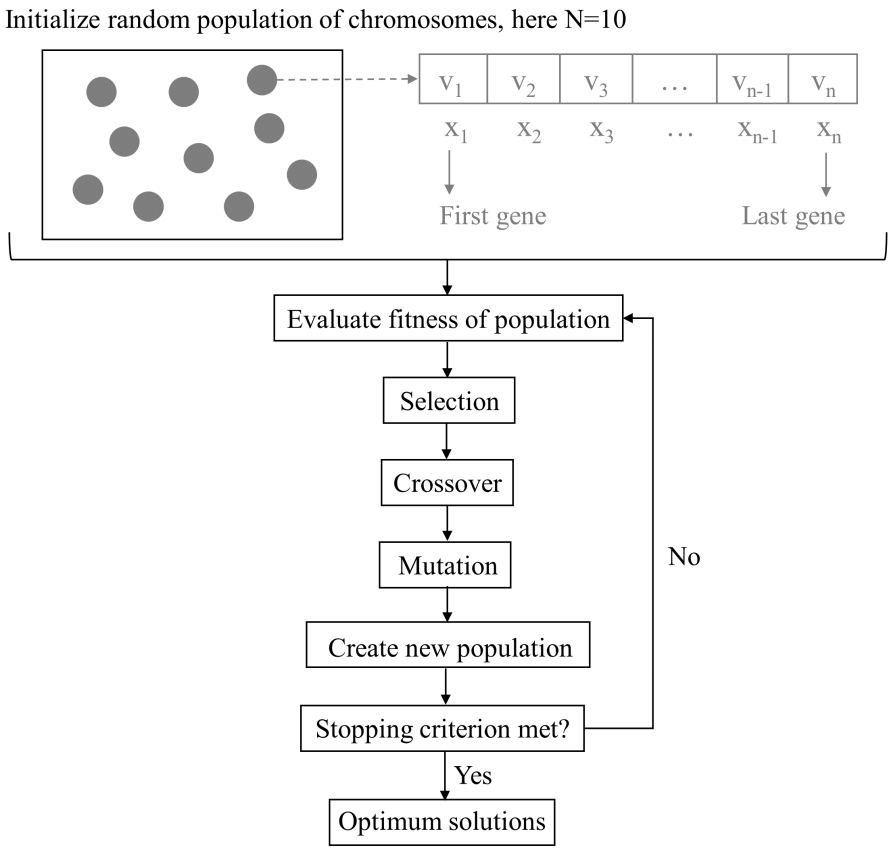

Figure 1: Illustration showing the general framework of a genetic algorithm.

nodes in real-world networks, resulting in the formation of a scale-free network [5]. A network is defined as $G = (V, E)$, where V is the set of nodes, and E is the set of edges, which can be either directed or undirected. Additionally, a network is considered complete if there exists an edge between every pair of vertices in V. To construct a network with N nodes using the BA algorithm, we begin with $m0$ nodes and connect them to form a complete network. The construction of a BA network is as illustrated in Algorithm 1. Subsequently, the remaining $N - m0$ nodes are added sequentially to simulate the network's growth. Each added node is connected to m edges, and the probability of the new node connecting to another node i is proportional to i's degree following eq. (2), where k_i is degree of node i.

$$p_i = \frac{k_i}{\sum_{n=1}^{N} k_n} \tag{2}$$

Through this process of preferential attachment, the BA algorithm can generate a scale-free network with a degree distribution following a power law. The power law distribution follows the formula $P(k) = k^{-\gamma}$, where k is the degree and γ is the degree exponent. In a scale-free network a few nodes, known as hubs, will have a significantly higher degree than the majority of nodes. Additionally, a scale-free network typically has a degree exponent $2 \leq \gamma \leq 3$, if $\gamma < 2$, the average degree diverges, and if $\gamma < 3$, the standard deviation of the degree diverges. This scale invariance makes scale-free networks lack a characteristic degree or scale, hence their name.

Algorithm 1 BA algorithm

```
function CreateNetwork(N, m0, m)
    G = Complete network of m0 nodes
    for i = 1 to N − m0 do
        Add node i to G
        for j = 1 to m do
            Add edge to node i using preferential
            attachment (2)
        end for
    end for
end function
```

3.4 Proposed approach

We propose an Enhanced Networked Genetic Algorithm (ENGA) in which we structure the population in the genetic algorithm using a scale-free network following the BA model, where hub nodes play a central role in driving the evolution of the population. Similar to GA, the first generation is randomly generated in the initialization phase. Subsequently, we create a scale-free network following the BA model (Algorithm 1). We calculate a score for each individual i in the population, reflecting both its genotypic and phenotypic importance, GI^i and PI^i, respectively. By matching the scores to the degrees of the nodes, we assign each individual to a node following the approach highlighted in Algorithm 2. The scores calculation depends on the algorithm's progress with more weight being attributed to the genotypic importance in the early stages. Consequently, diverse individuals receive higher scores, allowing the algorithm to explore the search space extensively. As the algorithm progresses, the weight shifts to the phenotypic side, associating higher scores with higher fitnesses. This shift enables the algorithm, in later stages, to focus on exploiting already explored regions in the search space. This dynamic approach to calculating individual scores achieves a better balance between exploration and exploitation, leading to improved results.

Formally, to calculate the scores of every individual, we first compute the similarity matrix of the population. The similarity between individual s^i and s^j is taken as the Euclidean distance between them, following the Equation (3):

$$Similarity(s^i, s^j) = \|s^i - s^j\| = \sqrt{\sum_{x=1}^{|X|}(v_x^i - v_x^j)^2} \tag{3}$$

To obtain the genotypic importance of node i, we average and normalize its similarity with all other nodes. Subsequently, we calculate the phenotypic importance of each node by applying the fitness function f and normalizing the values. We subtract 1 if the problem is a minimization problem to indicate that higher phenotypic importance is associated with lower fitness.

To combine genotypic and phenotypic importance, we calculate the corresponding weights based on the current generation. The algorithm starts with a phenotypic weight $W_{start} = 0.3$ and linearly grows to reach a maximum phenotypic weight of $W_{end} = 0.9$. This linear growth is based on the current generation gen and the maximum generation

gen_{max} following Equation (4):

$$W = W_{start} + (W_{end} - W_{start}) \times \left(\frac{gen}{gen_{max}} \right) \tag{4}$$

After calculating the weight, we combine the final score of an individual by applying

Algorithm 2 Population Assignment to Network

function ASSIGN($population, network, gen, gen_{max}$)
    Calculate similarity matrix (3)
    Calculate GI of each node
    Calculate PI of each node
    Calculate $W_{phenotype}$ for current generation (4)
    $W_{genotype}$ = 1 - $W_{phenotype}$
    for $i = 1$ to N **do**
        $score_i = W_{phenotype} * PI^i + W_{genotype} * GI^i$
    end for
    Assign a high-score individual to high-degree nodes
end function

the weight for each component and adding them. Following the calculation of the score for each individual, we associate high-scoring individuals with high-degree nodes. Since low-degree nodes are more likely to connect to higher-degree nodes, individuals with low scores will connect to individuals with higher scores. This dynamic connectivity ensures that hubs act as representatives, driving the evolution of the population and spreading high-quality information throughout the population.

For further clarification, the assignment process is illustrated in Fig. 2, where we present the assignment of 10 individuals based on their scores to a scale-free network. The size of the nodes is proportionate to their degree and the darkness of their colors is proportionate to the score of the individual assigned to them. It can be seen that larger nodes have darker colors, illustrating the association between a high score and a high degree as discussed above.

As presented in Algorithm 3, after the assignment of the first population, we evaluate the fitness of all nodes. We, then, proceed to select each node for mating and choose the second parent as a random neighbor of that node. Given that low-degree nodes are more likely to be connected to hubs, this allows hubs to control the exchange of important information in the population. The two chosen parents undergo crossover and mutation, giving rise to two new offspring. Subsequently, the offspring with the superior fitness is compared to the first parent. If the fitness of the offspring is better, it replaces the first parent. This strategy ensures that the algorithm avoids selecting individuals that could potentially lower the overall fitness of the population.

An overview of the ENGA process is presented in Fig. 3, highlighting each step of the algorithm and the network's evolution between two generations. The figure illustrates how the emergence of new chromosomes results in updated scores for all individuals, which induces a reshuffling of the chromosome-to-node assignments by mapping the updated scores to their corresponding node degrees. The depicted process is repeated until the last generation n_{max} is reached or the optimal value of the fitness function is discovered. With the emergence of each new generation, the similarity matrix and the scores of the chromosomes are recomputed and the individuals are reassigned to new nodes, resulting

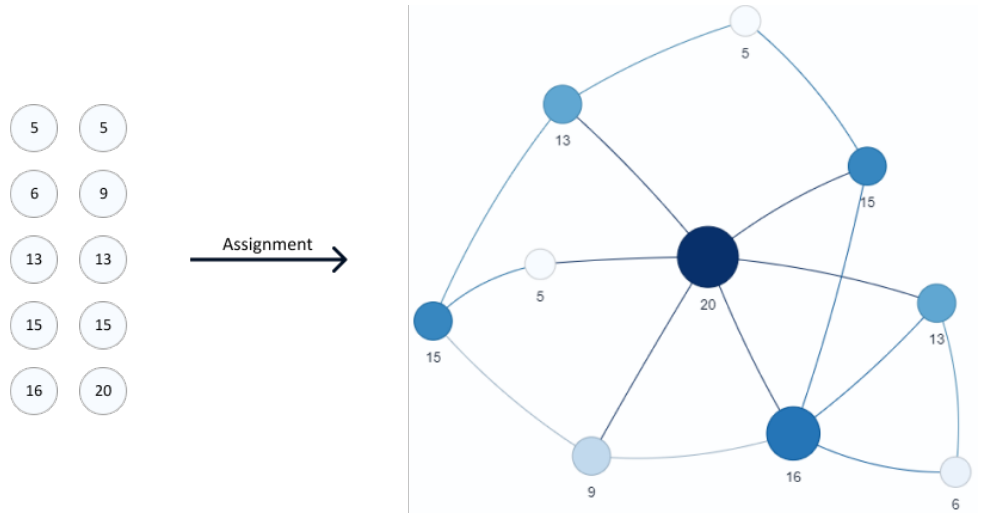

Figure 2: Illustration showing the assignment of 10 individuals based on their scores to a scale-free network

Algorithm 3 ENGA algorithm

function ENGA($N, m, m0, gen_{max}$)
    Create a random population of size N
    $CreateNetwork(N, m, m0)$ (1)
    $Assign(population, network, 0, gen_{max})$ (2)
    for $gen = 1$ to gen_{max} **do**
        for $parent_1$ in $nodes$ **do**
            $parent_2$ = Random Neighbor of $parent_1$
            $child_1$, $child_2$ = Crossover of the 2 parents followed by Mutation
            Get the best child based on fitness
            Replace $parent_1$ with the best child if better
        end for
        $Assign(population, network, gen, gen_{max})$ (2)
    end for
end function

in a constant flow between the nodes of the network across generations, as illustrated with the arrows in the figure.

3.5 Experimental setup

To assess the optimization performance of the proposed algorithm, a comprehensive comparative experiment is conducted on 8 widely-used benchmark functions [31, 16] presented in Table 1. We employ the same random initial population to evaluate the performances of different algorithms and we average the results of 50 independent runs considering the stochasticity inherent to evolutionary algorithms. All experiments are run on a 2.30 GHz Intel Core (TM) i7-12650H PC with 16 GB RAM and Windows 11 operating system. The benchmark functions we used were selected to include functions that display different features. In terms of modularity, functions f_1, f_2, f_6 and f_7 are unimodal, while all other functions are multimodal. The dimension of all problems is set to 30 and hence the number of genes in a chromosome is 30 for all benchmark functions.

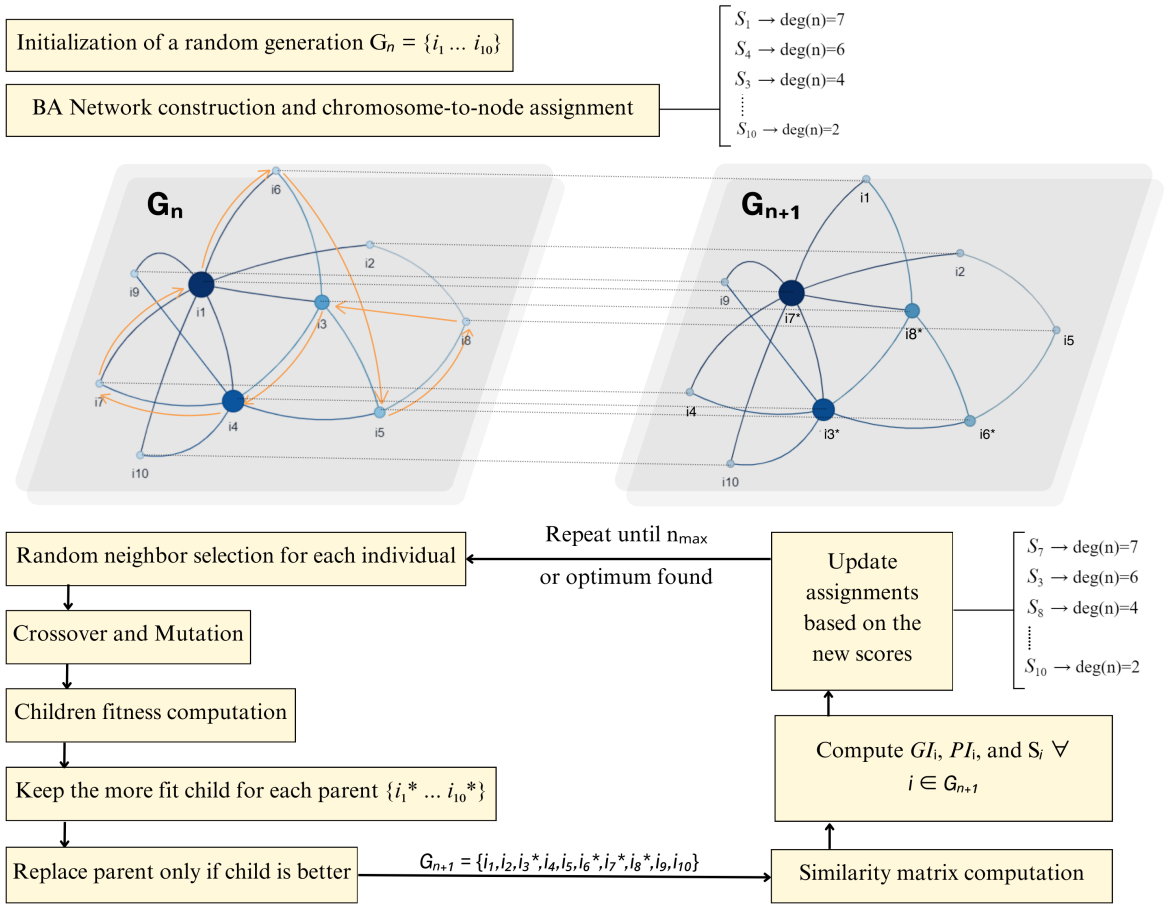

Figure 3: Illustration of ENGA framework on an exemplary network model of 10 nodes. The size and color contrast of the nodes are proportionate to their degrees. The offspring of the individuals of generation G_n are marked with an asterisk. The diagram highlights the evolution of the chromosome-to-node assignments based on the recomputed scores (S_i) of the chromosomes. The reshuffling of these assignments from G_n to G_{n+1} is marked with arrows.

Additionally, all genes in the chromosome have the same domain $D_1 = D_2 = \cdots = D_n$. It is noteworthy that $f2$ and $f3$ gene values have a decimal precision of 2, $f8$ has a decimal precision of 4, while all other functions have no decimal precision.

We compare the performance of ENGA with the traditional GA algorithm and the NGA algorithm [16] by assessing the average best fitness reached after 50 runs. Additionally, we present the standard deviation over the 50 runs. Since our algorithm performs optimally with a mutation rate of 0.2, and for a fair comparison with other algorithms, we conduct two experiments: one using a mutation rate of 0.1, as suggested by the authors of NGA [16], and another experiment with a mutation rate of 0.2. This ensures that even at a low mutation rate, ENGA outperforms other algorithms. We have adopted the parameters outlined in [16] for the GA to ensure a valid comparison. The specific parameters employed are summarized in Table 2.

Table 1: The benchmark functions used to assess the performance of the algorithms

ID	Function	Domain[1]
f1	$\sum_{i=1}^{30} v_i^2$	$[-100, 100]$
f2[2]	$\sum_{i=1}^{30} i \cdot v_i^4 + U[0,1]$	$[-1.28, 1.28]$
f3	$\sum_{i=1}^{30}(v_i^2 - 10\cos(2\pi v_i) + 10)$	$[-5.12, 5.12]$
f4	$\sum_{i=1}^{30} \frac{v_i^2}{4000} - \prod_{i=1}^{30} \cos\left(\frac{v_i}{\sqrt{i+1}}\right) + 1$	$[-600, 600]$
f5	$\sum_{i=1}^{29} 100(v_{i+1} - v_i^2)^2 + (v_i - 1)^2$	$[-32, 32]$
f6	$(v_1 - 1)^2 + \sum_{i=2}^{30}(i \cdot (2v_i^2 - v_{i-1})^2)$	[-10, 10]
f7	$10^6 v_1^2 + \sum_{i=2}^{30} v_i^2$	[-100, 100]
f8	$418.9829 \cdot 30 - \sum_{i=1}^{30} v_i \sin(\sqrt{\lvert v_i \rvert})$	[-500, 500]

[1] The domain represents the range of values that the variables in the corresponding benchmark function can take.
[2] $U[0,1]$ denotes a uniform distribution with values between 0 and 1.

Table 2: The configuration of operators and parameters in the algorithms.

Operators and parameters	Value
Population Size	125
Maximum Generation	2000
Crossover type	Single Point Crossover
Crossover rate	0.95
Mutation type	Single Point Mutation
m	4
m0	4

4 Results and analysis

In Table 3, we present the results of running all three algorithms GA, NGA, and ENGA on the functions presented in Table 1 with a mutation rate of 0.1. The mean best fitness values presented in Table 3 underscore the superior performance of the ENGA algorithm, particularly evident in functions $f7$ and $f8$, where ENGA reached the global optimum in $f7$ and came very close in $f8$. Notably, ENGA exhibits more effective exploitation of the global optimum region compared to GA and NGA. This outcome signifies the algorithm's adeptness at balancing exploration and exploitation, enabling it to navigate the search space efficiently. Furthermore, ENGA performs better in permutation variant functions $f5$ and $f6$. These functions are among the most challenging due to their dependence on the gene's position. Given a solution s, if we generate a permutation s' of s, then $f(s) \neq f(s')$. These functions trap the algorithm in many local optima, and escaping these local optima requires effective exploratory capabilities. As seen in the results, ENGA was able to create a balance between exploration and exploitation by using a scale-free network to model the population structure, considering the genotypic and phenotypic importance of each

Table 3: Average best fitness over 50 runs for each algorithm on the benchmark functions using a mutation rate of 0.1

Function ID	GA		NGA		ENGA	
	Mean	Std	Mean	Std	Mean	Std
f1	6.66E+0	2.36E+0	2.94E+0	1.63E+0	**2.00**E-**2**	1.39E-1
f2	8.60E-2	2.60E-2	3.14E-2	2.54E-2	**1.95**E-**2**	2.44E-2
f3	2.28E+0	7.39E-1	1.29E+0	3.08E-1	**1.76**E-**1**	7.27E-2
f4	1.05E+0	2.00E-2	1.01E+0	2.25E-2	**4.01**E-**1**	1.16E-1
f5	2.24E+2	2.98E+2	2.88E+2	3.65E+2	**1.64**E+**2**	2.92E+2
f6	3.22E+0	1.32E+0	9.10E+0	1.46E+1	**2.62**E+**0**	1.21E+0
f7	5.80E+0	2.62E+0	3.26E+0	1.35E+0	**0.00**E+**0**	0.00E+0
f8	2.10E+1	8.47E+0	9.88E+0	5.48E+0	**7.40**E-**1**	2.19E-1

node. To fully explore the potential of ENGA, we repeated the experiment with the same parameters in Table 2, but this time using a mutation rate of 0.2 for all algorithms. The ENGA mean best fitness values presented in Table 4 are significantly lower than those of GA and NGA. Notably, ENGA now achieves the global optimum in $f1$ with a mutation rate of 0.2. This enhanced performance in ENGA with the increased mutation rate can be attributed to the fact that the benchmark functions have a large domain for each value, making the search space more complex. The higher mutation rate, coupled with the balance between exploration and exploitation facilitated by the intragenerational links, allowed ENGA to increase its performance. Additionally, the replacement mechanism in ENGA, where we only replace the parent if its child is better, enables us to use a higher mutation rate without decreasing performance. Instead, it enhances performance by allowing more effective exploitation in the search space.

Without loss of generality and to illustrate, we plot the degree distribution of a network generated for function $f4$ in Fig. 4. It is clear that the network in ENGA is a scale-free network, and its degree distribution follows the power law. ENGA benefits from these hub nodes as the central source of information, as lower-degree nodes are more likely to be connected to these hubs. Thus, low-degree nodes allow for exploration based on the high-quality information found in the hub, while the hubs act as the central source of important information.

To further investigate the performance of ENGA, we provide a convergence performance comparison between the ENGA, NGA, and GA algorithms, as demonstrated in Fig. 5. It is apparent that the convergence rate of ENGA is superior to that of the NGA algorithm across all of the 4 benchmark functions $f2, f3, f4$, and $f5$. Additionally, in $f3$, the ENGA algorithm exhibits a very fast convergence speed. While fast convergence is not always necessarily advantageous as it may correlate with early convergence, in this case, it could be attributed to ENGA reaching the sub-search space of the global optimum faster, indicating superior exploration capabilities. Moreover, after reaching near the global optimum, the ENGA algorithm demonstrates efficient and faster exploitation compared to both NGA

Table 4: Average best fitness over 50 runs for each algorithm on the benchmark functions using a mutation rate of 0.2

Function ID	GA		NGA		ENGA	
	Mean	Std	Mean	Std	Mean	Std
f1	4.14E+0	1.98E+0	1.30E+0	1.37E+0	**0.00**E+**0**	0.00E+0
f2	1.38E-1	5.87E-2	3.32E-2	3.94E-2	**1.98**E-**2**	2.93E-2
f3	1.62E+0	6.38E-1	8.92E-1	3.26E-1	**3.76**E-**2**	2.07E-2
f4	1.02E+0	3.62E-2	9.59E-1	7.10E-2	**1.49**E-**1**	3.84E-2
f5	1.96E+2	1.92E+2	3.47E+2	6.47E+2	**1.41**E+**2**	2.95E+2
f6	3.20E+0	5.04E+0	2.3E+0	1.51E+0	**2.14**E+**0**	1.16E+0
f7	4.00E+0	2.13E+0	1.00E+0	1.26E+0	**0.00**E+**0**	0.00E+0
f8	1.39E+1	5.25E+0	4.35E+0	2.31E+0	**2.00**E-**1**	4.58E-2

and GA. This further illustrates how ENGA outperforms both NGA and GA in all the benchmark functions.

To ensure that ENGA is not suffering from early convergence and rapidly losing its genetic diversity, we investigated the evolution of the similarity matrix that is computed for each generation and compared it to that of NGA. To this end, we averaged the similarity matrices over 1000 generations starting from generation 200 for both ENGA and NGA. We excluded the first 200 generations as the genetic diversity is normally initially very high for both algorithms, which would significantly disrupt the scale of the plots and would not be relevant to the issue of early convergence. The results obtained for the function $f4$ could be found in the upper and lower left plots of Fig. 6. It is apparent that the average genotypic differences between the individuals tend to be significantly higher for NGA. We further transformed the similarity matrices to binary matrices in which all non-zero values are taken as 1. Subsequently, we summed the 1000 matrices for each of the algorithms and normalized them to the same scale. The resulting heatmaps are displayed in the right part of Fig. 6. These plots revealed that the counts of genotypic differences between the individuals of ENGA were almost twice as high as those of NGA. Overall, these heatmaps reveal that ENGA approaches the optimal value more rapidly and maintains diversity while efficiently exploiting the region around the optimal value. In contrast, NGA maintains less diversity and fails to approach the optimal value as efficiently as ENGA does.

Furthermore, to investigate the importance of the weighted phenotype and genotype components of the score used for individual-to-node assignment, we obtained the number of unique nodes in the population for all generations beyond generation 200 for ENGA, NGA, and a modified version of ENGA in which we neglected the genotypic component of the assignment score and made it exclusively dependent on the fitnesses of the individuals. Again, we exclude the first 200 generations as the number of unique nodes is naturally very high at the beginning and is not relevant to our investigation. It is well known that the number of unique nodes in the population significantly decreases in genetic algorithms as generations progress. The resulting plots displaying the evolution of the

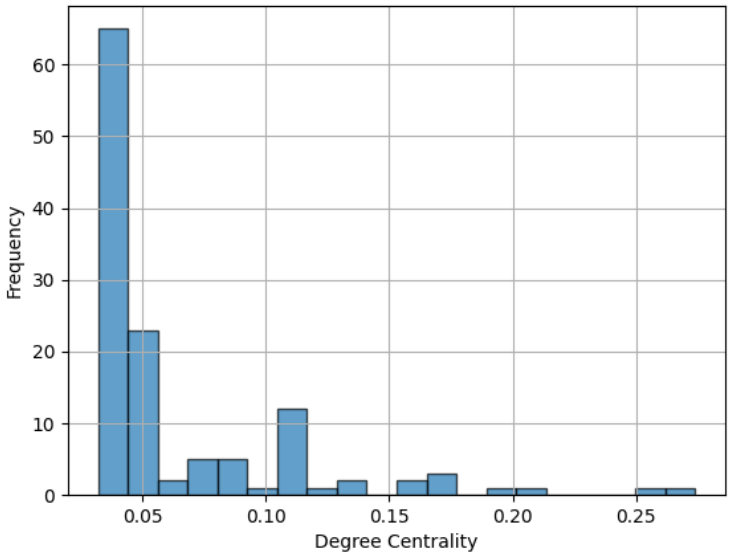

Figure 4: Illustration of the degree distribution of a network generated with ENGA for function $f4$

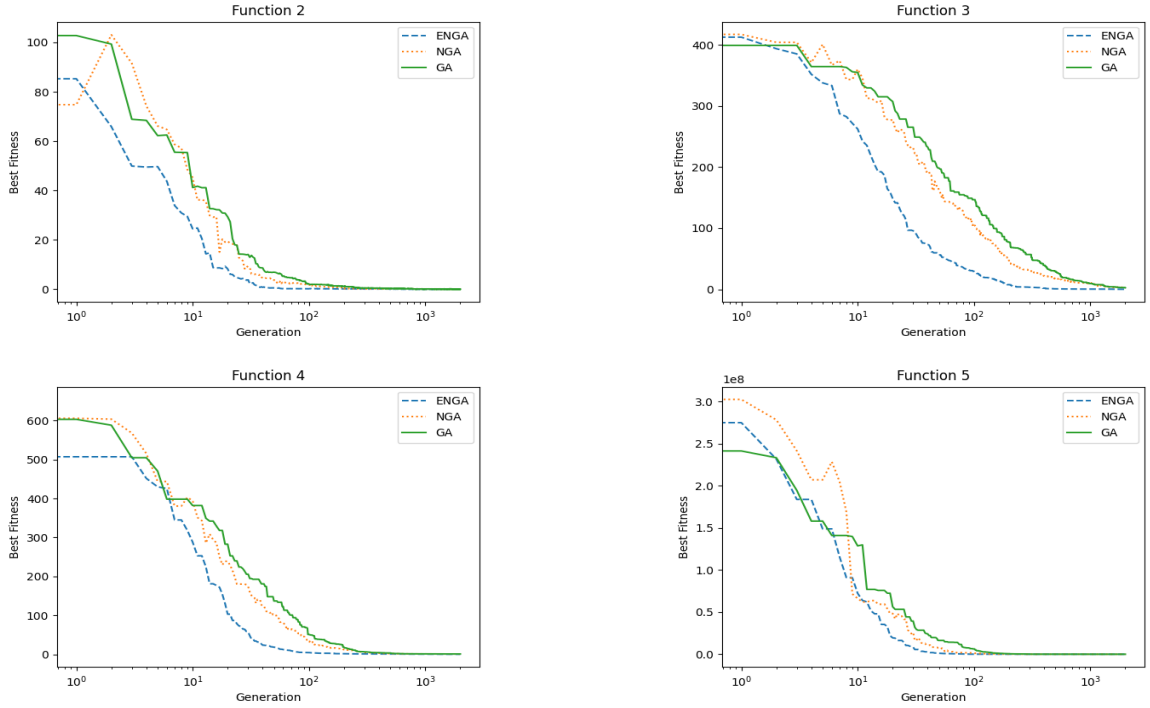

Figure 5: Convergence curves of GA, NGA, and ENGA on 4 benchmark functions.

number of unique nodes for the 3 algorithms for functions $f2$, $f3$, $f4$, and $f5$ are provided in Fig.7. These plots highlight the fact that neglecting the genotypic component of ENGA consistently results in rapid early convergence across all functions except $f2$, which displays an exceptional pattern since it has a uniformly random component. Yet, even for $f2$, the number of unique nodes for modified ENGA is found to be consistently significantly lower than that of ENGA. This confirms the validity of our motivation for using a hybrid

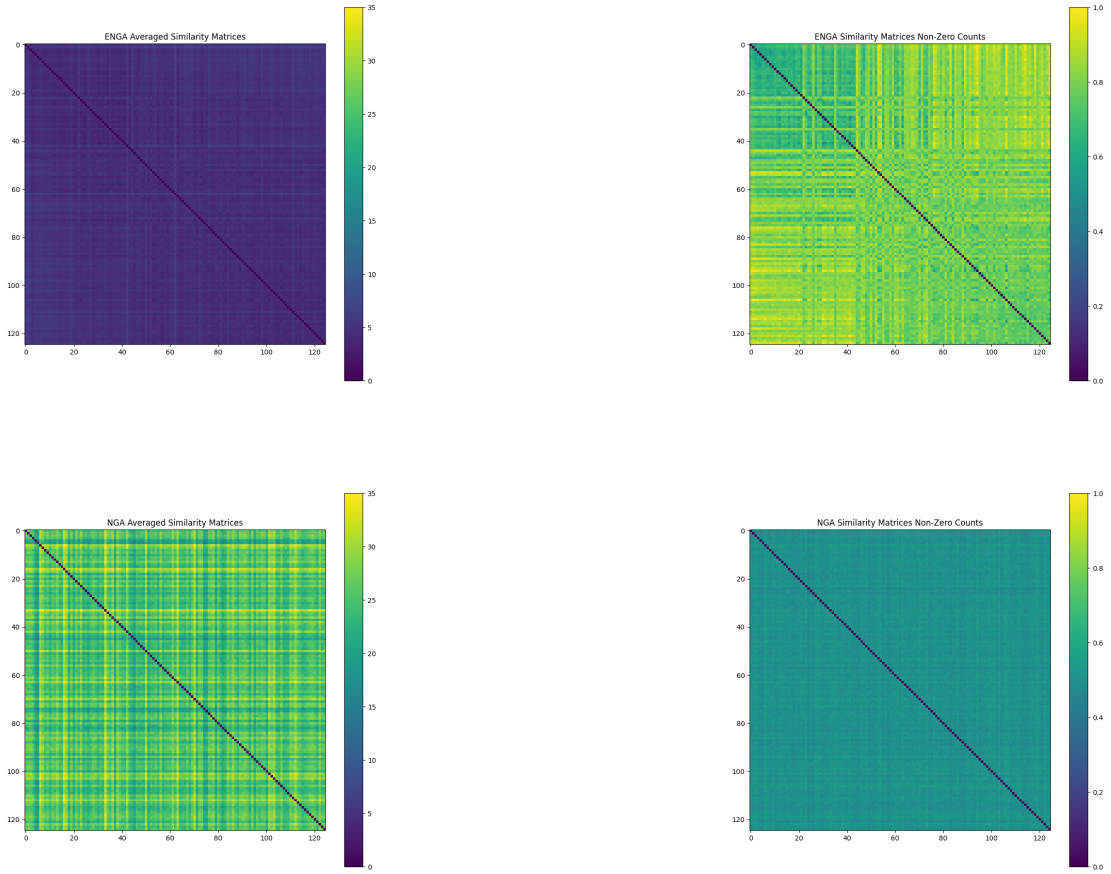

Figure 6: Heatmaps capturing the evolution of the similarity matrices of the individuals of NGA and ENGA for function $f4$ over 1000 generations starting with generation 200. On the left, the averaged similarity matrices for both algorithms are displayed with the same scale. On the right, the summed counts of all non-zero occurrences in the 1000 matrices are displayed. The counts are normalized with respect to the max value for both algorithms.

score that combines the phenotypic and genotypic components to avoid early convergence. Additionally, as previously discussed, we let the weights attributed to each of these two components evolve across generations to allow for a shift from exploration to enhanced exploitation. The effect of this evolution could be seen in the plots of functions $f3$ and $f4$, which both display a rapid increase in the number of unique individuals after the rapid initial decrease that results from approaching the optimal solution and marks the end of the exploration phase. This prompted increased results in a number of unique nodes that is significantly higher than that for both NGA and modified ENGA and that is maintained until the last generation. It is important to note that the evolution of the number of unique individuals varies depending on the function and the choice of the weights of the genotypic and phenotypic components of the assignment score. Accordingly, we initialize these weights as 0.7 for the genotype component and 0.3 for the phenotype one in order to initially promote exploration without neglecting the essential selective role to be played by the phenotypic component. Then, we let these weights linearly evolve to reach 0.1 and 0.9 at the last generation to shift towards an increased focus on exploitation without losing

the necessary genetic diversity.

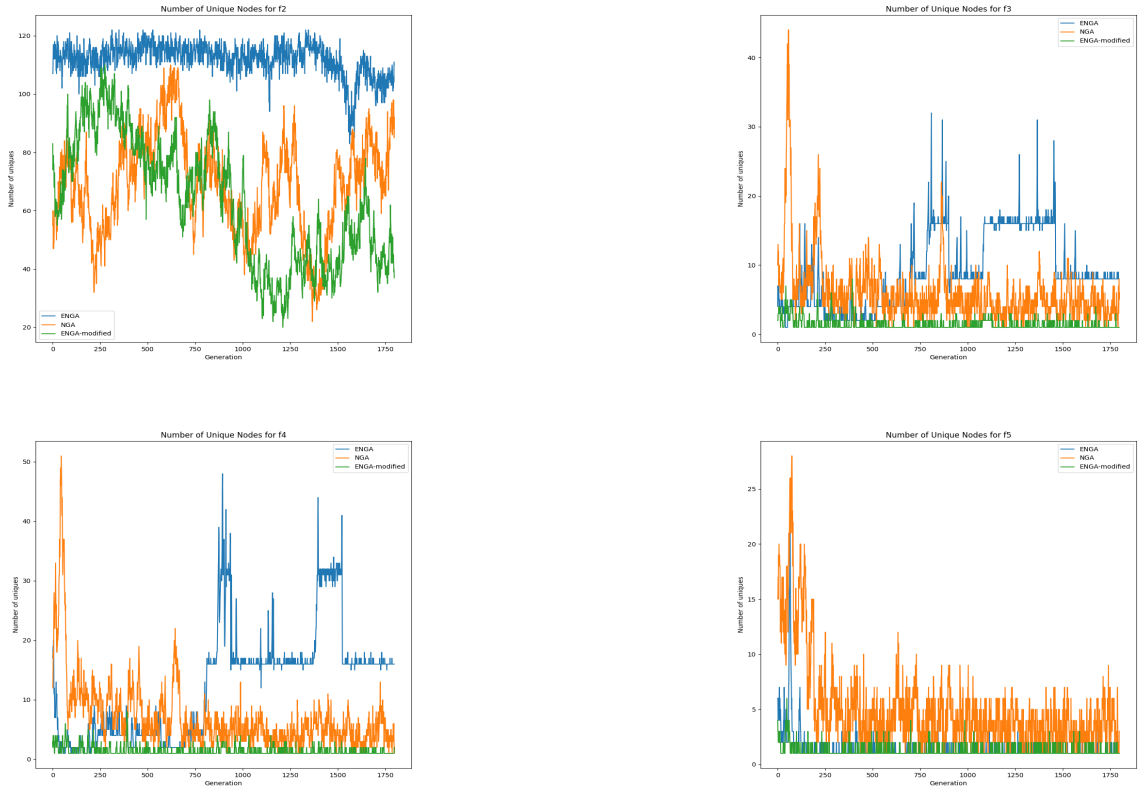

Figure 7: The frequency of unique nodes after generation 200 in the populations of ENGA, NGA, and modified ENGA with exclusive dependence on the phenotype component for functions $f2$, $f3$, $f5$, and $f6$.

Additionally, the plots of Fig.7 provide further confirmation of our previous analysis of the performance of ENGA and its capacity to maintain genetic diversity compared to NGA. For almost all functions, ENGA displays a higher number of unique nodes relative to NGA. This pattern holds for $f2$ for which ENGA is able to maintain a very high level of diversity until the last generation despite the deceptiveness introduced by the random component of the function. Moreover, these plots highlight that NGA tends to approach the different functions similarly consistently resulting in a pattern of unique individuals that oscillates rapidly within a small range of values. A similar pattern is observed for GA (not shown), yet at a higher level. Alternatively, ENGA proves its capacity to change its approach depending on the function, as illustrated by the different patterns it obtains in the plots of functions $f3$, $f4$, and $f5$. Moreover, further analysis revealed that it could obtain different patterns for the same function depending on the evolutionary path that happens to be followed. In the case of $f5$, ENGA will either rapidly reach the optimal value or will get stuck at one of 3 local optima until the last generation. It is impossible to break out of these local optima when using one-point crossover and single-point mutation as this requires the simultaneous change of the last 3 genes to reach the optimal value. This explains why ENGA displayed an exceptionally small number of unique nodes for $f7$ and maintained it so. Unlike NGA and GA, ENGA promotes further exploitation and increases the number of unique individuals only when such an increase could drive the evolutionary process in a better direction. This provides additional proof of the capacity

of ENGA to efficiently balance exploration and exploitation.

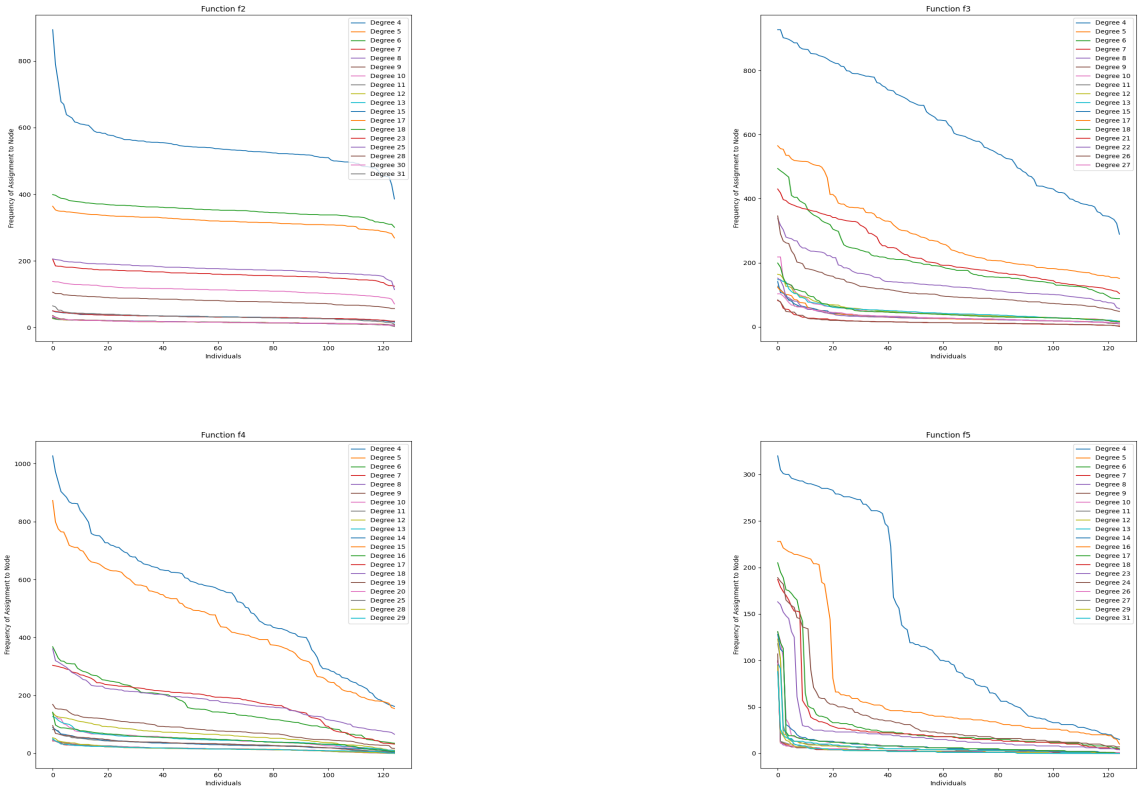

Figure 8: The frequency distributions of the assignment of individuals and their descendants to each category of nodes for functions $f2$, $f3$, $f5$, and $f6$.

Finally, we note that the proposed approach establishes an unprecedented perspective of network usage. The 125 individuals that are initially randomly generated are continuous across generations since the offspring of each of them maintains the same index across all generations. In other words, the child corresponding to each parent inherits its index. In parallel, the network remains unchanged while the individuals are repeatedly assigned to the nodes across the generations. This means that while the BA scale-free network we are using is not changing topologically, a constant flow between its nodes is maintained across the generations. Moving from generation i to generation i+1, the individual corresponding to each node is transmitted to another. Additionally, nodes with the same degree could be essentially considered the same, which allows us to reduce the network to a few categories of nodes that correspond to their degrees. Based on this vision, we could establish a frequency distribution for each category of nodes to capture the number of times each individual was assigned to it. To this end, we count the number of times each individual and all of its 124 descendants were assigned to each category of nodes and we provide the resulting frequency distributions for $f2$, $f3$,$f4$, and $f5$ in Fig.8. The resulting plots were found to follow a stable pattern over multiple runs of each function. The random component of $f2$ results in most individuals being equally assigned to the different categories of nodes resulting in the plateaus that could be seen across the distributions of the different categories of nodes. In contrast, $f3$ and $f4$ display a pattern of decay that starts linear for the low-degree nodes and gets more exponential as the node degree increases. This reflects that a few

individuals tend to be assigned to the higher degree nodes much more than all the others, which is consistent with the scale-free topology of the network. Function $f5$, on the other hand, consistently displays a very sharp decrease across the distributions of all node categories, which could be explained by the homogeneous pool of chromosomes that results once ENGA converges to either the optimal value or one of the 3 local optima previously discussed. These different function-dependent patterns of individual-to-node assignment capture the mechanism by which ENGA is able to obtain the previously discussed varying patterns at the level of the evolution of the counts of unique nodes.

5 Conclusion and Future Work

We highlighted the three main levels at which networks could enhance the performance of GA by establishing intra- or inter-generational communication routes between the evolved individuals. Our suggested approach focuses essentially on one of these aspects, namely the establishment of authority nodes to drive the evolutionary process. We benefit from the scale-free property of the BA model, which mimics real-world networks, to recruit these authority nodes. Moreover, we use a hybrid phenotype-genotype-based score to assign individuals to their corresponding nodes, and we dynamically evolve the weights attributed to the phenotype and genotype components of the score as well. This assignment approach and dynamic evolution of the weights allow us to establish a balance of exploration and exploitation proceeding from enhanced exploration to increased focus on exploitation. We demonstrate the superior explorative potential of ENGA as it is capable of approaching the optimal value faster than NGA and GA for the different benchmark functions. We further highlight that ENGA is able to do so while maintaining genetic diversity and that it efficiently balances exploration and exploitation as reflected by its capacity to prompt an increase in diversity and to do so only when this is likely to take the evolutionary process in a better direction. Moreover, we discuss how our use of a topologically stable network allows for the constant flow of individuals between its nodes, which follows stable patterns for the different functions.

The generation of one network to be used for all generations makes our algorithm computationally efficient and significantly faster than NGA. This computational efficiency is established without sacrificing the performance of the algorithm as ENGA outperformed both GA and NGA on all benchmark functions. While using the same scale-free network is beneficial in terms of computational efficiency, other benefits could be introduced by creating the scale-free network for each generation based on the hybrid phenotype-genotype scores. In addition, in our current approach, the weights for the phenotype and genotype are set to evolve linearly from 0.3 to 0.9 and 0.7 to 0.1, respectively. Alternatively, the specific choice of these values and their evolution could be made dependent on the evolutionary path that happens to be followed. We also note that considering the capacity of ENGA to rapidly approach the optimal value for all the benchmark functions we analyzed, its exploitative potential was demonstrated to a higher extent than its explorative potential. We hypothesize that testing ENGA on real-world applications will further reveal the superiority of its explorative and exploitative potentials. Moreover, despite our selection of benchmark functions with diverse features to avoid potential biases in testing the performance of ENGA, a few biases persisted. Notably, all the functions we ran our algorithm on had the same domain for all genes of their solution. Further work should explore the deviations in performance that could result from testing ENGA on cases in which different genes have different domains.

Another important modification that could enhance our design is to make the network dynamic, allowing it to evolve across generations based on the evolutionary process. Alternatively, other non-BA static networks could be used instead. For instance, a static network with communities could be used to constantly segregate the population of solutions into distinct communities. Additionally, multiple static nodes could be used together to account for several topological effects simultaneously in a multi-layered fashion. For example, separate networks could be used to account for the genotypic and phenotypic components. Nodes would be assigned to each of these two layers which will be used together to drive the mating of the individuals and thus the evolutionary process. Finally, intergenerational links could be established between several networks or the same network to account for the transmissions across nodes undergone by each individual. In this approach, two nodes in different generations could be linked when an individual is assigned to one and then to the other.

References

[1] Enrique Alba, Hugo Alfonso, and Bernabé Dorronsoro. Advanced models of cellular genetic algorithms evaluated on sat. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1123–1130, 2005.

[2] Enrique Alba and Bernabé Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE transactions on evolutionary computation*, 9(2):126–142, 2005.

[3] AM Algelany, MA El-Shorbagy, et al. Chaotic enhanced genetic algorithm for solving the nonlinear system of equations. *Computational Intelligence and Neuroscience*, 2022, 2022.

[4] Bogdan Artyushenko. Analysis of global exploration of island model genetic algorithm. In *2009 10th International Conference-The Experience of Designing and Application of CAD Systems in Microelectronics*, pages 280–281. IEEE, 2009.

[5] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

[6] Mohamed A Belal and Mohamed H Haggag. A structured-population genetic-algorithm based on hierarchical hypercube of genes expressions. *International Journal of Computer Applications*, 64(22), 2013.

[7] Ms Trupti Bhoskar, Mr Omkar K Kulkarni, Mr Ninad K Kulkarni, Ms Sujata L Patekar, GM Kakandikar, and VM Nandedkar. Genetic algorithm and its applications to mechanical engineering: A review. *Materials Today: Proceedings*, 2(4-5):2624–2630, 2015.

[8] Katy Börner, Soma Sanyal, Alessandro Vespignani, et al. Network science. *Annu. rev. inf. sci. technol.*, 41(1):537–607, 2007.

[9] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64:1695–1724, 2013.

[10] Edmund K Burke, Steven Gustafson, and Graham Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004.

[11] Erick Cantu-Paz. Efficient and accurate parallel genetic algorithms, 2000.

[12] Robert J Collins and David R Jefferson. Selection in massively parallel genetic algorithms, 1991.

[13] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, 45(3):1–33, 2013.

[14] Dipankar Dasgupta and Zbigniew Michalewicz. Evolutionary algorithms in engineering applications, 2013.

[15] Ke-Lin Du, MNS Swamy, et al. Search and optimization by metaheuristics. *Techniques and Algorithms Inspired by Nature*, pages 1–10, 2016.

[16] Wenbo Du, Mingyuan Zhang, Wen Ying, Matjaž Perc, Ke Tang, Xianbin Cao, and Dapeng Wu. The networked evolutionary algorithm: A network science perspective. *Applied Mathematics and Computation*, 338:33–43, 2018.

[17] Felipe P Espinoza, Barbara S Minsker, and David E Goldberg. A self adaptive hybrid genetic algorithm. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 759–759, 2001.

[18] Paulo M França, Jatinder ND Gupta, Alexandre S Mendes, Pablo Moscato, and Klaas J Veltink. Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers & Industrial Engineering*, 48(3):491–506, 2005.

[19] Kerry Gallagher and Malcolm Sambridge. Genetic algorithms: a powerful tool for large-scale nonlinear optimization problems. *Computers & Geosciences*, 20(7-8):1229–1236, 1994.

[20] Andrea Gasparri, Stefano Panzieri, and Federica Pascucci. A spatially structured genetic algorithm for multi-robot localization. *Intelligent Service Robotics*, 2:31–40, 2009.

[21] Ali Ghaheri, Saeed Shoar, Mohammad Naderan, and Sayed Shahabuddin Hoseini. The applications of genetic algorithms in medicine. *Oman medical journal*, 30(6):406, 2015.

[22] Mario Giacobini, Mike Preuss, and Marco Tomassini. Effects of scale-free and small-world topologies on binary coded self-adaptive cea. In *Evolutionary Computation in Combinatorial Optimization: 6th European Conference, EvoCOP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings 6*, pages 86–98. Springer, 2006.

[23] Deepti Gupta and Shabina Ghafir. An overview of methods maintaining diversity in genetic algorithms. *International journal of emerging technology and advanced engineering*, 2(5):56–60, 2012.

[24] W Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, 1990.

[25] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[26] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

[27] Michael Kirley and Robert Stewart. Multiobjective evolutionary algorithms on complex networks. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 81–95. Springer, 2007.

[28] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91(9):992–1007, 2006.

[29] Thiemo Krink, Brian H Mayoh, and Zbigniew Michalewicz. A patchwork model for evolutionary algorithms with structured and variable size populations. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1321–1328, 1999.

[30] Daekyung Lee and Beom Jun Kim. Different environmental conditions in genetic algorithm. *Physica A: Statistical Mechanics and its Applications*, 602:127604, 2022.

[31] Jing J Liang, Bo Y Qu, and Ponnuthurai N Suganthan. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 635(2), 2013.

[32] Ting Yee Lim. Structured population genetic algorithms: a literature survey. *Artificial Intelligence Review*, 41:385–399, 2014.

[33] Holger R Maier, Saman Razavi, Zoran Kapelan, L Shawn Matott, J Kasprzyk, and Bryan A Tolson. Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental modelling & software*, 114:195–213, 2019.

[34] John McCall. Genetic algorithms for modelling and optimisation. *Journal of computational and Applied Mathematics*, 184(1):205–222, 2005.

[35] Rui Mendes, James Kennedy, and José Neves. The fully informed particle swarm: simpler, maybe better. *IEEE transactions on evolutionary computation*, 8(3):204–210, 2004.

[36] Rohan Mukherjee, Gyana Ranjan Patra, Rupam Kundu, and Swagatam Das. Cluster-based differential evolution with crowding archive for niching in dynamic environments. *Information Sciences*, 267:58–82, 2014.

[37] Joshua L Payne, Mario Giacobini, and Jason H Moore. Complex and dynamic population structures: synthesis, open questions, and future directions. *Soft computing*, 17:1109–1120, 2013.

[38] Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on evolutionary computation*, 14(5):782–800, 2010.

[39] Dobrivoje Popovic and KCS Murty. Retaining diversity of search point distribution through a breeder genetic algorithm for neural network learning. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 1, pages 495–498. IEEE, 1997.

[40] Mike Preuss and Christian Lasarczyk. On the importance of information speed in structured populations. In *International Conference on Parallel Problem Solving from Nature*, pages 91–100. Springer, 2004.

[41] Ralf Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.

[42] Dirk Schlierkamp-Voosen and Heinz Mühlenbein. Strategy adaptation by competing subpopulations. In *Parallel Problem Solving from Nature—PPSN III: International Conference on Evolutionary Computation The Third Conference on Parallel Problem Solving from Nature Jerusalem, Israel, October 9–14, 1994 Proceedings 3*, pages 199–208. Springer, 1994.

[43] Kit-Sang Tang, Kim-Fung Man, Sam Kwong, and Qianhua He. Genetic algorithms and their applications. *IEEE signal processing magazine*, 13(6):22–37, 1996.

[44] Saneh Lata Yadav and Asha Sohal. Comparative study of different selection techniques in genetic algorithm. *International Journal of Engineering, Science and Mathematics*, 6(3):174–180, 2017.

[45] Raed Abu Zitar, Mohammed Azmi Al-Betar, Mohammed A Awadallah, Iyad Abu Doush, and Khaled Assaleh. An intensive and comprehensive overview of jaya algorithm, its versions and applications. *Archives of Computational Methods in Engineering*, 29(2):763–792, 2022.