Stratified Opposition-Based Initialization for Variable-Length Chromosome Shortest Path Problem Evolutionary Algorithms [⋆]

Aiman Ghannami[1,*], Jing Li[1], Ammar Hawbani[1], Ahmed Al-Dubai[2]

Abstract

Initialization is the first and a major step in the implementation of evolutionary algorithms (EAs). Although there are many common general methods to initialize EAs such as the pseudo-random number generator (PRNG), there is no single method that can fit every problem. This study provides a new, flexible, diversity-aware, and easy-to-implement initialization method for a genetic algorithm for the shortest path problem. The proposed algorithm, called stratified opposition-based sampling (SOBS), considers phenotype and genotype diversity while striving to achieve the best fitness for the initialization population. SOBS does not depend on a specific type of sampling, because the main goal is to stratify the sampling space. SOBS aims at an initial population with higher fitness and diversity in the phenotype and genotype. To investigate the performance of SOBS, four network models were used to simulate real-world networks. Compared with the most frequently used initialization method, that is, PRNG, SOBS provides more accurate solutions, better running time with less memory usage, and an initial population with higher fitness. Statistical analysis showed that SOBS yields solutions with higher accuracy in 68%–100% of the time. Although this study was focused on the genetic algorithm, it can

[⋆]This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (A Class) No. XDA19020102.

[*]Corresponding author

Email addresses: aiman@mail.ustc.edu.cn (Aiman Ghannami), lj@ustc.edu.cn (Jing Li), anmande@ustc.edu.cn (Ammar Hawbani), a.al-dubai@napier.ac.uk (Ahmed Al-Dubai)

[1]University of Science and Technology of China,JinZhai 96, Hefei,China.

[2]University of Edinburgh,South Bridge, Edinburgh EH8 9YL, United Kingdom.

be applied to other population-based EAs that solve the shortest path problem and use the same direct population representation such as particle swarm optimization (PSO).

Keywords: Shortest Path Problem; Initialization ; Genetic Algorithm; Network Kriging;

1. Introduction

A genetic algorithm (GA) for the shortest path problem (SP) (Ahn & Ramakrishna, 2002) aims to find the shortest path between a pair of nodes in a given network. There are many applications of the SP in different fields, including routing communication networks (Moy, 1994), transportation (Zhan & Noon, 1998), robotic path planning (Desaulniers & Soumis, 1995), and the design of very large-scale integrated circuits (Cong et al., 1998). A GA is typically a population-based stochastic search algorithm. The first step of the search process is to have an initial guess: the initialization population. The quality of the initial population is related directly to the quality of the solution obtained by the GA. A *good* initial guess gives the GA a better chance of finding a good solution (Burke et al., 2004) (Rahnamayan et al., 2007)(Clerc, 2008), and a *bad* initial guess may hinder the ability of the algorithm to find the optimal solution (Maaranen et al., 2004). The definition of a *good* and *bad* initial population is a problem-specific matter, as there is no general rule of initialization that can be applied to every type of problem (Lunacek & Whitley, 2006). In general, according to (Kazimipour et al., 2014), the initialization methods in evolutionary algorithms (EAs) can be classified into three main groups.

1. Randomness: a random method that generates different individuals using a pseudo-random number generator (PRNG) (Kazimipour et al., 2014).

2. Generality can be divided into two main groups: *generic* and *application-specific* techniques.

3. Compositionality, whereby the number of standalone procedures used to generate initial individuals is one or more. It is considered *non-compositional*

if it consists of more than one step, such as opposition-based learning (OBL) (Tizhoosh, 2005); otherwise, it is considered *compositional.*

Throughout the GA SP literature, random initialization (using a PRNG) is used, as suggested by (Ahn & Ramakrishna, 2002). PRNG is commonly used because the problems solved with EAs are black-box optimization problems (Kazimipour et al., 2012) (i.e., there is no prior knowledge regarding the search landscape). This is also applicable to the GA SP, where there is no prior information about the length/cost of the optimal path. However, in this type of searching problems, it is impossible to have the best initial guess (Rahnamayan et al., 2008), but it is possible to have a better guess using the OBL (Tizhoosh, 2005) concept. The basic idea of OBL is to obtain complementary candidate solutions for a set of solutions. As most EAs start by randomly guessing the solutions, OBL suggests guessing a solution and its "opposite." According to OBL, the opposite of number x can be calculated using $\breve{x} = a + b - x$. Fig. 1 illustrates the relationship between x and its opposite, $\breve{x}$, in interval $[a, b]$ according to OBL. There are two ways of employing OBL in searching: defining a function that maps every solution to its opposite solution; searching for solutions with opposite characteristic(s) (Rojas-Morales et al., 2017). The latter approach is used in this study, based on the path length as the main solution characteristic.

This study presents a stratified opposition-based sampling (SOBS) initialization method for the GA SP with variable-length chromosomes. SOBS aims to obtain the initial population with a *good* fitness and high length-level diversity without ignoring genotype diversity.

The remainder of this paper is organized as follows. Section 2 provides a review of the GA SP applications and initialization algorithms in EAs. Section 3 introduces the proposed algorithm, followed by the experiment and the simulation results in Section 4. The initial population diversity metrics and results are presented in Section 5. Section 6 details the statistical tests. Section 7 explains the effect of the repair function on the length-level diversity. Section 8 concludes this paper.

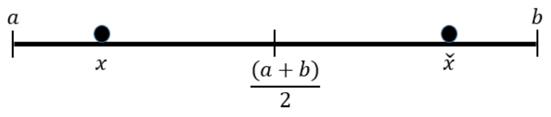


Figure 1: OBL concept (Tizhoosh, 2005)

2. Related Work

2.1. GA SP Applications

Even though GAs with variable-length chromosomes have found widespread applications in many fields, the research on this field is unsatisfactory (Qiongbing & Lixin, 2016). There are a few examples of studies on GAs with variable-length chromosomes, including most of the work in the last four years. In traffic and congestion planning, the GA SP is one of the most widely used algorithms (Mohamed et al., 2018). It has been used in vehicle routing optimization in (Miao et al., 2018)(Tiwari & Chang, 2015) (Khankhour et al., 2019). In irrigation systems, the GA SP is used to optimize pipe network paths (Zhao et al., 2019). In urban planning, the GA SP is used in pedestrian navigation systems (Zhou et al., 2019), personal navigation systems (Maruyama et al., 2004), and optimizing road networks in transportation (Chen et al., 2018). In cyber-physical power systems, it has been used for risk assessment (Dong et al., 2018) as a quick method that can reconstruct alternative routes. In(Chen et al., 2020) (Shafiee & Berglund, 2016), the GA SP was used in rescue planning to optimize the road selection between fire stations and destinations to identify the shortest routes for emergency vehicles to minimize the tour time and cost. In (Chen et al., 2017), the GA SP was used for urban areas for rescue planning in disasters such as earthquakes. In (Ayo et al., 2017), the GA SP was used in air transportation as a quick method to suggest flight paths when rerouting was needed. In (Vignesh & Premalatha, 2019), the GA SP was used in military information system optimization. In (Balta et al., 2017), the GA SP was used in small batch manufacturing to connect users with small batch production needs.

In (Vignesh & Premalatha, 2019), the GA SP was used in computer networks to create an energy-aware routing algorithm for software-defined networks, and in (Rastgoo & Sattari-Naeini, 2018), it was used to create a quality-of-service (QoS)-aware routing method. To the best of our knowledge, all of these applications use random initialization, except (Rastgoo & Sattari-Naeini, 2018), in which a problem-specific method that relies on an underlying QoS routing protocol was used to initialize the GA. Moreover, to the best of our knowledge, no previous work has reported a modification in the algorithm design, except (Qiongbing & Lixin, 2016), in which a new crossover method was proposed.

2.2. EA Initialization Methods

PRNG is the most commonly used method in EA initialization(Elsayed et al., 2016). PRNG generates a sequence of numbers that characteristically approximates a sequence of random numbers, but is not truly random (Kazimipour et al., 2014). That is, it depends on a set of initial seeds to generate the sequence (Park & Miller, 1988). The popularity of PRNG stems from the fact that it is simple and uniform. In addition, it can be easily implemented because it is available with any programming language (Kazimipour et al., 2014). The main problem with PRNG is that it suffers from the curse of dimensionality (Maaranen et al., 2004). In the SP, the number of feasible solutions can be very high. For example, the number of feasible solutions for source–destination pair 1 and 20 in the topology shown in Fig. 2 can be as high as 301600 (i.e., the total number of simple paths between nodes 1 and 20). Moreover, PRNG generates points that are not distributed properly (Kazimipour et al., 2014)(Helwig & Wanka, 2008). A chaotic number generator (CNG) studies the behavior of dynamic systems and uses chaos theory (Schuster & Just, 2006) to produce a stochastic initialization population. Sensitivity to initial conditions/populations is a common characteristic between dynamic systems and EAs; thus, CNGs have been adopted in many EAs (Ozer, 2010)(Dong et al., 2012)(Gao & Wang, 2007)(Akay & Karaboga, 2012)(Gao et al., 2012)(Gutiérrez et al., 2011). To produce a chaotic-like initialization population, the CNG needs a map. The mapping function in its basic form is as follows.

$$x_{(i,j)}^{G} = f(x_{(i,j)}^{G}) \tag{1}$$

Here, $x_{(i,j)}^{G}$ is the j^{th} variable of the i^{th} individual at the G^{th} generation. Mapping function f is not limited to one type; different functions can be employed, such as circle, sinus, ten, and logistic (Ozer, 2010). Mapping functions of CNGs are designed for problems with one to three decision variables (Senkerik et al., 2013), making the chaotic property advantageous in problems with low dimensionality. Compared with the aforementioned stochastic techniques, deterministic methods (also known as low-discrepancy methods (Uy et al., 2007)) do not prioritize unpredictability or randomness, and they prioritize the coverage of the search space. That is, the population should cover the entire search space (Kazimipour et al., 2013). Regardless of the initial seed, deterministic techniques generate the same population. Quasi-random sequence (QRS) and uniform experimental design (UED) are both deterministic initialization techniques. Despite its name, QRS is a completely deterministic method (Maaranen et al., 2007). QRS assumes that the objective function value and the initial population discrepancy are positively correlated, making it difficult to satisfy this assumption in practice (Kazimipour et al., 2014). Moreover, QRS assumes that the initial population covers the entire searching space, and this is difficult to satisfy in high-dimensional problems (Kazimipour et al., 2013), similar to the SP. Furthermore, QRS requires a measurement algorithm to measure discrepancies (Wang & Sloan, 2008). In addition, QRSs were originally designed for continuous spaces. UED is a type of space-filling technique (WangYuan & KaiTai, 1981), which searches for points that are distributed uniformly in a specific range. UED defines a D-dimensional array with q different levels and then searches a population size of q^{D}. Theoretically, q is a perfectly uniform population; thus, it is impossible to feed an algorithm with a population of this number to be evaluated. This is because the population should be sufficiently large to cover the entire array and increase exponentially.

3. Proposed Method

This section introduces the proposed algorithm. First, we derive a phenotype space probability stratification analysis, followed by the interpretation of genotype space probability stratification.

3.1. Stratifying Phenotype Space

In a given graph, G, many edges are traversed by many paths. For example, taking vertices v_1 and v_{20} in Fig. 2 as the source and destination, respectively, paths $\{1 \rightarrow 4 \rightarrow 10 \rightarrow 19 \rightarrow 20\}$ and $\{1 \rightarrow 4 \rightarrow 10 \rightarrow 15 \rightarrow 20\}$ traverse the same two edges, $(1, 4)$ and $(4, 10)$. As these paths share edges, these common edges among paths are reflected in path costs. This concept is well known in network kriging and may be used to obtain end-to-end estimation on a graph (Chua et al., 2006).

Moreover, the degree of vertex v_1 is $deg(v_1) = 4$; therefore, as long as the randomly initialized population number, n, is > 4, then at least one edge is traversed by at least two paths (by the Pigeonhole principle). For destination vertex v_{20}, $deg(v_{20}) = 3$; therefore, if $n > 3$, then at least one edge(s) is traversed by at least two paths. Taking the initial population number $n = V$, where V is the total number of vertices in the network, then each edge connected to source v_1 is traversed by h paths, where $1 \leq h \leq 17$ and the n initial paths terminate at the same destination (vertex v_{20}), with the edges of v_{20} traversed by $1 \leq h \leq 18$ paths.

Consequently, the difference in path costs reflects the cost of edges that are *not* shared (Hand, 2010). This difference can be measured using a string metric. Because paths vary in length, the Levenshtein distance (Wagner & Fischer, 1974) is a suitable metric as it provides an accurate pairwise and order-aware measurement of the distance between a pair of strings. The recursive function of the Levenshtein distance between two given chromosomes, x and y, with lengths of $|x|$ and $|y|$, respectively, is defined as follows. (Levenshtein,

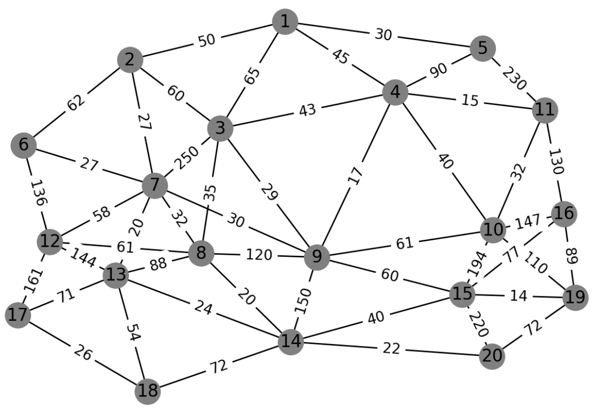


Figure 2: Typical 20-node random network (Ahn & Ramakrishna, 2002)

1966)

$$lev_{x,y}(i,j) = \begin{cases} max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev_{x,y}(i-1,j) + 1 \\ lev_{x,y}(i,j-1) + 1 \\ lev_{x,y}(i-1,j-1) + 1_{(x_i \neq y_j)} \end{cases} & \text{otherwise} \end{cases} \tag{2}$$

Here, $1_{(x_i \neq y_j)}$ is the characteristic function equal to 0 when $x_i = y_j$ (i.e., nodes i and j are the same) and equal to 1 otherwise.

The inputs of Algorithm 1 are the N population from which the initial population is taken, the sample size (population size) and the number of individuals are taken from strata . Those N individuals are then stratified according to their length into M strata (line 1), and the resulting strata are sorted according to the length of the individuals they contain (line 3). Algorithm 1 begins by checking whether the population size is already satisfied (line 8); if not, the leftmost and rightmost strata are chosen in lines 11–12. Before taking a sample of size θ, the stratum size is evaluated to ensure that θ is not larger than the number of individuals in the stratum (lines 13 and 18). In the next iteration, Algorithm 1 chooses the next leftmost and next rightmost stratum (lines 11–12) to select new individuals, and this continues until the population size is met.

Note that it does not matter whether the strata are ascending or descending. In addition, it is important to have sampling without replacement. If one iteration is not sufficient to obtain the required population number, we may have a second iteration (a complete iteration is when all strata have been visited). In this case, to ensure sampling without replacement, after adding the selected individuals to the initial population (lines 23–24), those individuals should be removed (lines 25–26). Finally, Algorithm 1 increases the strata index, C, (line 28) to ensure that we visit the next leftmost and rightmost strata. Thus, Algorithm 1 stratifies the phenotype space by stratifying paths according to their length. The maximum intra-stratum Levenshtein distance is limited to the length of paths in that stratum, as follows.

Given a sample space, S, with N individuals stratified according to their length into M disjoint subsets (strata) such that Ω_k, $k = 1, 2, 3, \ldots, M$, with $\bigcup_{k=1}^{M} \Omega_k = S$ and $\Omega_p \cap \Omega_q = \emptyset$; $p \neq q$.

Let each stratum be identified by the path/individual(s) length it contains (e.g., Ω_i is the stratum containing individuals of length i). These strata are sorted by their identifier such that $\Omega_i < \Omega_{i+1} < \cdots < \Omega_{m-1} < \Omega_m$, where Ω_i is the stratum that contains the shortest individual(s), and Ω_m is the stratum with the longest individual(s).

Let $lev_{k,k}^{max}$ be the maximum Levenshtein distance between any given paths, $\mathcal{P}_x$ and $\mathcal{P}_y$, in a given stratum, Ω_k. Then, $lev_{k,k}^{max}$ is as follows:

$$lev_{k,k}^{max} = lev(\mathcal{P}_x, \mathcal{P}_y) = k - 2, \tag{3}$$

where k is the length of the individuals in Ω_k. The maximum number of different edges between any two given paths in Ω_k is $lev_{k,k}^{max} + 1$. Constant 2 in Eq. (3) is used to exclude the source and destination vertices. According to the problem encoding, every chromosome starts with the source node and terminates with the destination node (i.e., the GA SP uses direct encoding/representation; there is no mapping between genotype and phenotype spaces (Rothlauf, 2006). A node can be referred to as a gene in this context. Consequently, paths and chromosomes are different names for the same entity. Throughout the paper, we use the terms chromosome, solution, and path interchangeably). Thus, the minimum common vertices between any given two paths is $min(\mathcal{P}_x \cap \mathcal{P}_y) = 2$.

In contrast, in the unstratified space, $lev_{i,m}^{max}$ is obtained for any given path, $\mathcal{P}_i$ and $\mathcal{P}_m$, in strata Ω_i and Ω_m, respectively, as follows:

$$lev_{i,m}^{max} = lev(\mathcal{P}_i, \mathcal{P}_m) = max(length(\mathcal{P}_i, \mathcal{P}_m)) - 2 = m - 2, \tag{4}$$

with $(lev_{i,m}^{max} + 1)$ of different edges.

Using an intuitive distance-based explanation of OBL for one-dimensional space (Rahnamayan et al., 2012), we can replace the Euclidean distance of (Rahnamayan et al., 2012) by the Levenshtein distance as explained previously. Then, SOBS obtains an initial population with better fitness compared with pure PRNG as follows.

We assume the following.

(a) We know nothing about the cost/length of the optimal solution, x_p. This is a black-box optimization problem. In addition, there exists one optimal solution for the problem.

(b) Let each stratum be identified by the chromosome(s) length it contains. These strata are sorted by their identifier such that $\Omega_i < \Omega_{i+1} < \cdots < \Omega_{m-1} < \Omega_m$, where Ω_i is the stratum that contains the shortest chromosome(s), and Ω_m is the stratum with the longest chromosome(s).

(c) The opposite of $x_i \in \Omega_i$ is defined as $x_m \in \Omega_m$, the opposite of $x_{i+1} \in \Omega_{i+1}$ is defined as $x_{m-1} \in \Omega_{m-1}$, and so on, such that $\Omega_i < \Omega_{i+1} < \cdots < \Omega_{m-1} < \Omega_m$, where i is the length of the solutions that belong to stratum Ω_i.

(d) Let $Dis_{x,y}$ be the number of the different edges between two given solutions, x and y, in strata Ω_x and Ω_y, respectively. It is defined as $Dis_{x,y} = lev(\mathcal{P}_x, \mathcal{P}_y) + 1$, where $lev(\mathcal{P}_x, \mathcal{P}_y)$ is the Levenshtein distance between $\mathcal{P}_x$ and $\mathcal{P}_y$.

(e) Let $\Omega_{\tilde{x}}$ be the median strata as in Fig. 3.

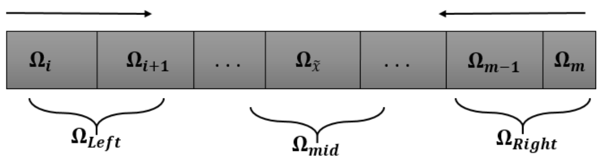


Figure 3: Illustration of sorted strata and SOBS sampling

To concisely exhaust the possibilities for the location of x_p, we refer to the strata on the left as Ω_{Left} and those on the right as Ω_{Right}, as illustrated in Fig. 3. Now, we have the following expected scenarios.

1. Algorithm 1 has visited Ω_{Left} strata and $\Omega_p \in \Omega_{Left}$; then, intuitively, Dis_{x_p,x_L} is minimized. That is, the maximum Levenshtein distance between the unknown optimal solution, x_p, and a selected solution, x_L, is minimized. Similarly, if $\Omega_p \in \Omega_{Right}$ and Ω_{Right} have been visited, Dis_{x_p,x_R} is at its minimum. Moreover, suppose $x_p \in \Omega_{m-1}$ and strata Ω_m and Ω_{m-2} have been visited. Because Ω_m and Ω_{m-2} are the direct neighbors of Ω_{m-1}, the possible maximum distance between any solutions from these strata and x_p is minimized from both sides. For instance, if Ω_{m-2} and Ω_m in Fig. 3 contain solutions with lengths of 8 and 10, respectively, then the length of x_p is 9. In this case, $Dis_{m-2,m-1} = 9 - 2 + 1 = 8$ and $Dis_{m,m-1} = 10 - 2 + 1 = 9$.

2. Owing to the constraints on the size of the initial population, Algorithm 1 has stopped before visiting Ω_{mid}. In this case, as explained previously, the possible maximum distance is minimized from both sides because Ω_{Left} and Ω_{Right} have been visited. Another important factor is that the lengths in the middle can be produced by the repair function. This is because the repair function job removes loops (i.e., shrinks solutions) and completes the repaired chromosomes by acting directly on the topology. The experimental verification of this effect of the repair function is described in Section 7.

Algorithm 1: GA SP initialization (SOBS)

```
Input: $N$ Population, Sample_size, $\theta$ number individual(s) per stratum.
Output: $n$ sample of population (Pop).
1  Strata ← stratify(N);// Stratify chromosomes
2  $M$ ← len(Strata);// Get the number of strata
3  Sort(Strata);
4  Pop←[ ];
5  C←0;
6  while $C <$ Sample_size do
7      LeftStratum,RightStratum,LeftSample,RightSample=[ ];
8      if length (Pop)= Sample_size or length (Pop)=$M$ then
9          Return Pop;
10     end
11     LeftStratum ← Strata[C];// Get lefmost stratum population
12     RightStratum ← Strata($M$-C);// Rightmost stratum
13     if length (LeftStratum)$\geq$ 2 then
14         LeftSample ← (pick(LeftStratum, $\theta$));// pick $\theta$ individuals
15     else
16         LeftSample ← (pick(LeftStratum,1));// Add a single
               individual
17     end
18     if length (RightStratum)$\geq$ 2 then
19         RightSample ←(pick(RightStratum, $\theta$));
20     else
21         RightSample ← (pick(RightStratum,1));
22     end
23     Pop.add(LeftSample);
24     Pop.add(RightSample);
25     LeftStratum.del(LeftSample);// Delete picked individuals
26     RightStratum.del(RightSample);
27     C ← C+1;
28 end
```

3.2. Stratifying Genotype Space

Section 3.1 shows how stratifying paths by lengths reduces variance in terms of phenotype diversity and how OBL can be used to minimize the difference (described by Levenshtein distance) between the initial guess and the optimal solution. In terms of genotype diversity, and by sampling from the opposite extreme side strata, Algorithm 1 moves the sampling pointer from the stratum with the fewest genes per chromosome to the stratum with most genes per chromosome. By definition, a feasible solution is a solution without any repeated genes (i.e., a path with no loops). Thus, in terms of genotype diversity, a chromosome from Ω_i is less diverse than a chromosome from Ω_{i+1}. Formally, let div_i be the genotype diversity of any path $\mathcal{P}_i$, $\mathcal{P}_i \in \Omega_i$, and let div_{i+1} be the genotype diversity of $\mathcal{P}_{i+1} \in \Omega_{i+1}$, and so on, where i is the length of chromosomes in Ω_i, then

$$div_i < div_{i+1} < \cdots < div_{m-1} < div_m. \tag{5}$$

Parameter θ in Algorithm 1 can be viewed as an initial population genotype diversity tuner. Inherently from Eq. (2) and Eq. (3), the larger the sample from a stratum with individuals of longest length (Ω_{Right}), the greater the genotype diversity in the initial population. As diversity is directly related to exploration (Gupta et al., 2006), more diversity means more exploration time. If $\theta = 1$, then we obtain a more diverse population than if $\theta = 2$ (assuming sampling without replacement).

4. Experiment

Unless stated otherwise, the algorithm parameters used in this section are listed in Table 1. The fitness function used is as follows (Ahn & Ramakrishna, 2002).

$$f_i = \frac{1}{\sum_{j=1}^{l_i-1} C_{g_i}(j), g_i(j+1)} \tag{6}$$

Here, f_i is the fitness of the ith chromosome, l_i is its length, $g_i(j)$ is the gene/node in the jth locus of the ith chromosome, and C is the link cost between nodes. The objective is to maximize f_i. For the detailed design of the GA SP and the parameters in Table 1, refer to (Ahn & Ramakrishna, 2002).

Table 1: Algorithm parameters

Selection type	Tournament selection
Selection pressure	2
Mutation rate	0.01
Crossover	1-point crossover
Population size	Number of nodes
Sampling	Random
Elitism	None
Number of generations	100

4.1. Results of the Deterministic Network

The first test involves a fixed 20-node network, with the topology illustrated in Fig. 2. The population size is the same as the number of nodes in the network. The chosen source and destination nodes are 1 and 20, respectively. The dashed line in Fig. 4. shows the optimal solution cost obtained by using the Dijkstra algorithm. Fig. 4. shows SOBS versus PRNG with 1000 samples. Each experiment was terminated at 100 generations. The box plots depict the spread of the obtained data. The lines inside the boxes denote the median. The upper line of the box is the upper quartile, and 75% of the obtained values fall below this line. The lower line of the box represents the lower quartile, and 25% of the data fall below this line. Based on the description given previously, it is obvious that most of the obtained values of SOBS are closer to the optimal solution fitness (dashed line). The closer the obtained values are to the optimal solution, the better.

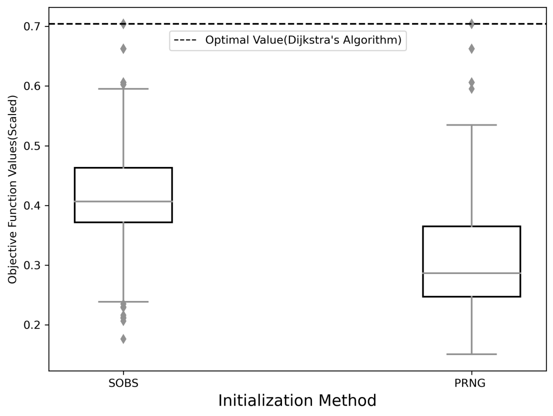


Figure 4: Summary of the best values achieved using the topology in Fig. 2 between nodes 1 and 20

4.2. Random Networks

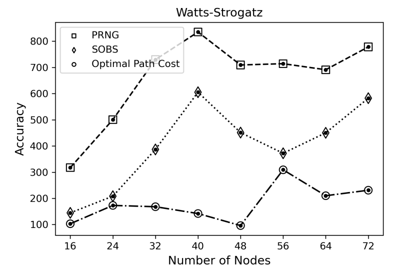
In this test, four models are used to generate different types of random networks. The first and second models are Watts–Strogatz (WS) for small-world networks, and Erdős—Rényi (ER) is used to generate general random networks. Numerous network systems exhibit small-world network properties such as social networks, neuroscience, medicine and bio-engineering (She et al., 2016), biochemical pathways (Telesford et al., 2011), and professional footballers' movements (Gama et al., 2015). Parameters in the ER(n, p) model were set to $p = 0.2$ for every random network generated, where n is the number of vertices and p is the probability of edge creation. For WS(n, k, p), $k = 4$ and $p = 0.1$, where n is the number of vertices, k is the mean that each vertex is connected with its k nearest neighbors in a ring topology, and p is the probability of rewiring each edge.

The third model is the Waxman model, which is commonly used to simulate topologies that resemble communications networks and intra-domain networks and to evaluate routing performance (Van Mieghem, 2001)(Verma et al., 1998)(De Neve & Van Mieghem, 2000)(Shaikh et al., 2001)(Guo & Matta, 2003).

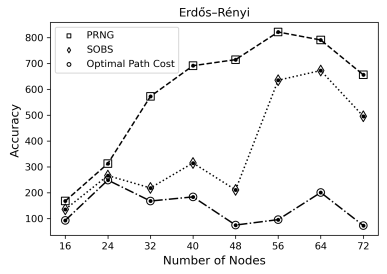
The parameter values of the Waxman model, Waxman(n, α, β), used in this experiment were set as $\alpha = \beta = 0.4$, where n is the number of vertices and α and β are the model parameters.

Finally, the fourth model is the Barabási–Albert (BA) model (Barabási & Albert, 1999) used to generate scale-free networks. Parameter m in BA(n, m) was set as $m = 2$, where n is the number of vertices and m is the number of edges to attach from a new vertex to existing vertices.

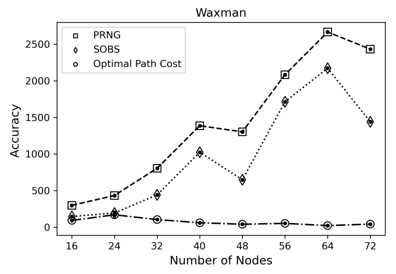
In each model, a network size of V, $V \in \{16, 24, 32, 40, 48, 56, 64, 72\}$, was generated with random integer weights between 1 and 100 for each edge. For each V, a random pair of source–destination was chosen; then the algorithm ran for 100 generations (to prevent the algorithm from converging before 100 generations, the optimal path was removed if found in the initial population), and the returned best solution achieved was recorded. This was repeated 1000 times to obtain 1000 samples. The mean of the best solution achieved over 1000 samples is shown in Fig. 5, where the x-axis represents the path costs $({f_i}^{-1})$, and the lower the obtained path cost, the better. The algorithm parameters in this experiment are listed in Table 1. The value of θ in Algorithm 1 was set as $\theta = 2$.



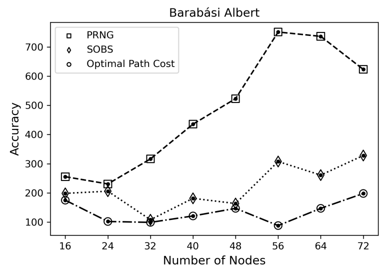
(a) Watts–Strogatz model



(b) Erdős—Rényi model



(c) Waxman model



(d) Barabási–Albert model

Figure 5: Accuracy of PRNG and SOBS with different network sizes. Accuracy is the mean cost of the best solution achieved over 1000 samples for each network size, with a constant number of generations of 100.

For every random network generated, the population size for PRNG was set to the number of network nodes. For the proposed initialization method, according to Algorithm 1, it cannot exceed the number of network nodes, but can be less than that if all the strata have been visited and sampled. Fig. 5 shows the means of the best solutions achieved against the optimal values. In all models and for each network size, SOBS solutions were closer to the optimal values than were the solutions of PRNG. Fig. 6 shows the fitness means of the initial population for each model. That is, the fitness of all individuals in the initial population was evaluated using fitness function f_i, and the mean value was selected and depicted in Fig. 6. The higher the obtained values, the better. Each violin plot presents 8000 data samples of the fitness mean across the eight different network sizes. Wider sections of the violin plots represent a higher probability that the data points take on the given x-axis value. Thus, the density of fitness means shows that SOBS has a higher probability of having higher (better) fitness values.

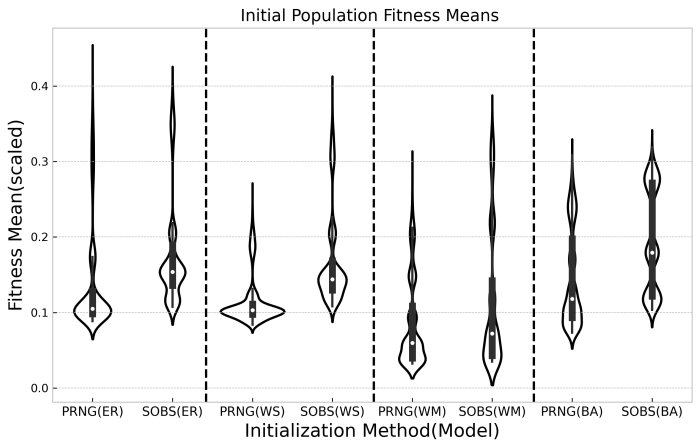
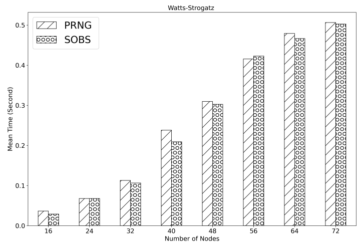
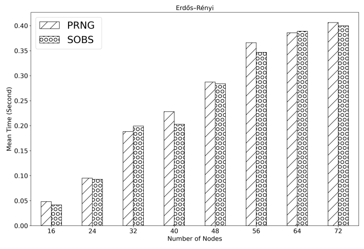


Figure 6: Initialization population fitness mean over 8000 samples (1000 samples for each network size)

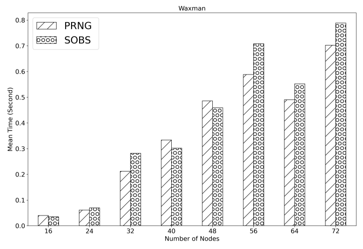
In terms of the running time (initialization time is included), Fig. 7 illustrates the average running time over 1000 samples for each network size.



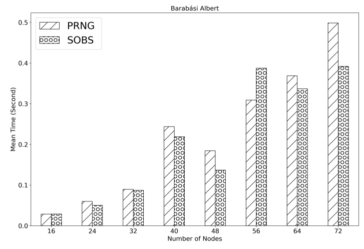
(a) Watts–Strogatz model



(b) Erdős—Rényi model



(c) Waxman model



(d) Barabási–Albert model

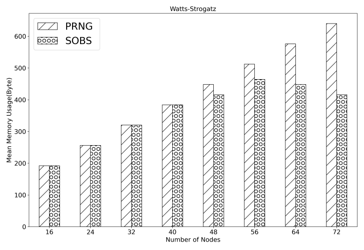
Figure 7: Mean execution time over 1000 samples

Although the SOBS initialization method strives for diversity in terms of gene-level diversity by selecting the longest chromosomes, it does not allow too much diversity. Too much diversity in EAs means too much exploration (Gupta et al., 2006), resulting in a longer searching time that hinders the ability of the algorithm to reach a better solution in a small number of generations.

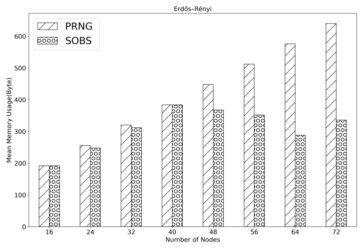
The memory usage of all models and the network size are shown in Fig. 8. Although the population number in many networks was the same (SOBS population number is reported in Table 2; for PRNG, the population number is the number of nodes in the network), except in a few cases, the memory usage was less than that of PRNG in many network sizes. This is because longer chromosomes occupy more memory locations than shorter chromosomes do. When the number of longer chromosomes is dominant in the sample space, they dominate the initial population in PRNG. In contrast, SOBS is not affected by the dominance of one length because the sampling inside any stratum with respect to the chromosome length is uniform.

Table 2: SOBS initial population number for each network model/size (note that for PRNG, the size of population is set to the network size)

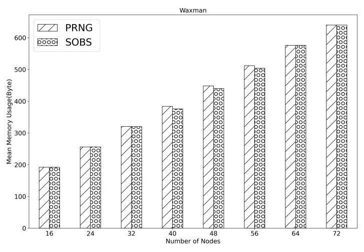
Network Size	Watts–Strogatz	Erdős—Rényi	Waxman	Barabási–Albert
16	16	16	16	15
24	24	23	24	23
32	32	31	32	32
40	40	40	39	40
48	40	38	47	47
56	56	36	55	56
64	64	28	64	64
72	72	34	72	71



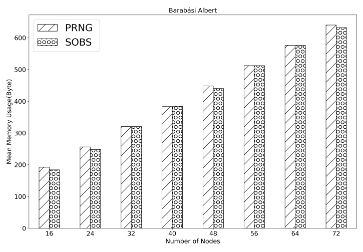
(a) Watts–Strogatz model



(b) Erdős—Rényi model



(c) Waxman model



(d) Barabási–Albert model

Figure 8: Mean of memory usage of the initial population, with sample size of 1000

5. Initial Population Diversity Comparison

This section details the experimental verification of the diversity values for both initialization methods. The goal was to compare the diversity of the initial population in terms of genotype and chromosome length level. First, we introduce the diversity metrics used to implement this test, and then the tested diversity results are reported.

5.1. Length-Level Diversity Metrics

For length-based diversity, we used two metrics. The first was used to measure the *richness* in terms of chromosome lengths. Richness is the number of different lengths available in the initial population. The greater the number of different lengths, the richer is the initial population in terms of length. Formally, given a sampling space, S, with N chromosomes satisfied into M different strata according to their homogeneous length, such that $\Omega_k,\ k = 1, 2, 3, \ldots, M$, with $\bigcup_{k=1}^{M} \Omega_k = S$ and $\Omega_p \cap \Omega_q = \emptyset$, $p \neq q$, then richness ϑ is

$$\vartheta = M. \tag{7}$$

To measure diversity among the available lengths, the Simpson index (Magurran, 2013) was chosen. The Simpson index provides a measurement of dominance among a group of populations. The dominance measurement indicates whether the population is dominated by one or more types. Evenness among the population can also be inferred from the same metric, as dominance and evenness are complementary factors. That is, it indicates whether the population is dominated by some lengths. The Simpson index of diversity is given by

$$D = 1 - \sum \frac{n(n-1)}{N(N-1)}, \tag{8}$$

where n is the total number of a particular type (length) and N is the total number of all types (lengths). For more clarification of how D is applied on the length level, please refer to Algorithm 2. The first line in Algorithm 2 takes the set of lengths as an input. Line 3 counts the total number of lengths, and the remaining lines are the formula in Eq. (8).

Algorithm 2: Calculating the Simpson diversity index on chromosome lengths.

```
Input: A set of the available lengths in the initial population, $L$.
Output: $D$ in Eq. (8).
1  NumeratorList ← [ ];
2  Numerator ← 0;
3  $L$_sum ← Sum($L$);
4  for $i \leftarrow 0$ to $len(L)$ do
5      n ← $L$[i];
6      NumeratorList.append (n*(n-1));
7  end
8  Numerator ← Sum (NumeratorList);
9  Denominator ← ( $L$_sum*( $L$_sum -1));
10 $D$ ← 1-((Numerator)/Denominator);
11 Return $D$ ;
```

5.2. Initial Population Genotype Diversity Metric

For gene-level diversity, the Levenshtein distance was used. As a pairwise distance metric between chromosomes x and y, the Levenshtein distance is defined as follows.

$$L_d = \frac{2 * \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} NLD_{(x,y)}}{N(N-1)} \tag{9}$$

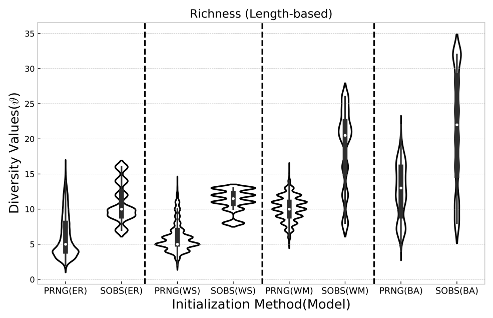
Here, N is the size of the initial population and $NLD_{(x,y)}$ is the normalized Levenshtein distance, calculated as follows.

$$NLD_{(x,y)} = \frac{Lev_{(x,y)}}{Max(length(x), length(y))} \tag{10}$$

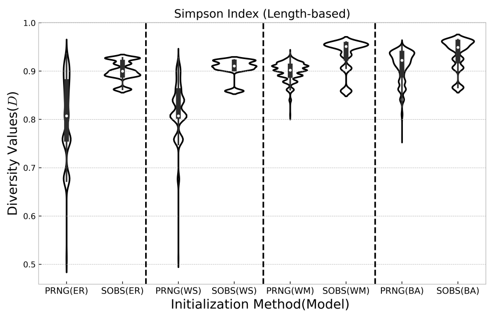
5.3. Experimental Results of Diversity

Fig. 9 shows diversity at the length and gene levels. The gene-level diversity is measured by the pairwise Levenshtein Distance in Eq. (8). Each violin plot in Fig. 9 depicts the distribution of 8000 samples of diversity values (1000 samples for each network size). For the length-level diversity, SOBS provided a richer

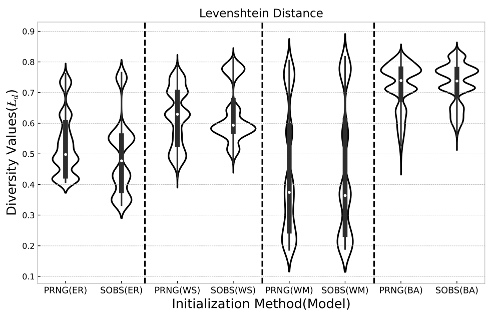
(Fig. 9(a)) and more diverse (Fig. 9(b)) initial population, even with a smaller population size, as indicated by the results in Table 2. In terms of genotype diversity, Fig. 9(c) shows SOBS diversity values as well as those of PRNG. In the ER and WS models, SOBS has less diversity, whereas in the Waxman and BA models, the diversity value distribution is similar to that of PRNG.



(a) Richness (length-based diversity)



(b) Simpson index (length-based diversity)



(c) Genotype diversity (Levenshtein distance)

Figure 9: Diversity comparison of both initialization methods with each network model, where each violin plot contains 8000 samples

6. Statistical Tests

6.1. Verifying Eq. (3) and Eq. (4)

To recap, Eq. (3) and Eq. (4) in Section 3 suggest that, when sampling from strata where each stratum has only paths of the same length, the variability in terms of costs of those paths should be less than the variability of paths that are not stratified (i.e., as in pure PRNG). A dispersion measurement on intra-stratum and inter-stratum levels can provide a good indication of this variability. For example, the topology shown in Fig. 2. We can select the first and the last nodes, nodes 1 and 20, then exhaust all the possible solutions, count every solution cost, and then measure the dispersion of all solutions. Then, the solutions are stratified, cost of each solution is counted, and dispersion of each stratum (intra-stratum) is measured. Because the number of strata can be large (for example, after exhausting all the solutions between nodes 1 and 20 in Fig. 2, the number of strata is 16), to present the results in a comprehensible manner, we decided, after measuring the dispersion of each stratum, to report only the maximum and minimum dispersion values.

Exhausting the search space between every source and destination pair in a graph is very computationally expensive and time-consuming. Moreover, it is difficult to report the results of such a large number of possibilities in a table. A good way to solve this problem is to choose a source–destination pair and use their search space for the measurement. To measure dispersion, the interquartile range (IQR) measure of variability was used. The IQR is resistant to outliers and skewness of data. Table 3 presents the results of this test, and Algorithm 3 shows the detailed implementation of this test. The first entry in the table is dedicated to the topology shown in Fig. 2. with the source and destination nodes, 1 and 20, respectively. With the other models, a generated random network with a network size of 20 nodes was chosen, and the parameters of each model are presented in Section 4. From Table 3, we can see that, with stratification, we can always obtain paths that are closer to the optimal solution, in terms of their costs, than those that can be obtained in the case of an unstratified space being used (i.e., the maximum IQR of intra-stratum is less than the IQR of inter-stratum). This means that the SOBS selects a population that is more similar to the optimal solution than PRNG does, as expressed by Eq. (3) and Eq. (4).

Table 3: Dispersion (IQR) results among costs of paths at the intra-stratum and inter-stratum level.

Network type	Intra-stratum IQR		Inter-stratum IQR	S size	Number of strata
	Min	Max			
Deterministic (Fig. 2)	108	289	359	301600	16
WS model	38	118	212	44216	15
ER model	11	114	169	116592	17
Waxman model	0	115	126.25	1192	13
BA model	0	107	208	1433	16

Algorithm 3: Statistical test to verify Eq. (2) and Eq. (3).

```
Input: $S$; List of all solutions in the search space between a pair of
       Src-Dst.
Output: Min_Intra_StratumIQR, Max_Intra_StratumIQR,
        Inter_StartaIQR.
1  Inter_StartaIQR $\leftarrow$ 0;
2  Max_Intra_StratumIQR $\leftarrow$ 0;
3  Min_Intra_StratumIQR $\leftarrow$ 0;
4  Inter_StrataCosts $\leftarrow$ [ ];
5  Intra_StratumCosts $\leftarrow$ [ ];
6  Strata_IQR $\leftarrow$ [ ];
7  Strata $\leftarrow$ { } ;// A dictionary, keys are strata IDs, and values are paths
     of each stratum.;
8  for $i \leftarrow 0$ to $len(S)$ do
9      Inter_StrataCosts.Append(Cost(S[$i$])); // Obtain the cost of each
         path;
10 end
11 Inter_StartaIQR=IQR(Inter_StrataCosts);//Obtain Inter-stratum
     dispersion;
12 Strata $\leftarrow$ Stratify $S$ ;
13 for $j \leftarrow 0$ to $len(Strata)$ do
14     Paths$\leftarrow$ Strata[j] ; //Get all the paths of each stratum separately.;
15     Intra_StratumCosts$\leftarrow$ [ ];
16     for $p \leftarrow 0$ to $len(Paths)$ do
17         Intra_StratumCosts.Append(Cost(Paths[$p$]));
18     end
19     Strata_IQR.Append(IQR(Intra_StratumCosts));
20 end
21 Max_Intra_StratumIQR $\leftarrow$ max(Strata_IQR); //Maximum IQR ;
22 Min_Intra_StratumIQR $\leftarrow$ min(Strata_IQR);
```

6.2. Significance Tests

To compare the results of both algorithms, Vargha and Delaney statistics and Mann–Whitney U test were used based on the practical guidelines provided by (Arcuri & Briand, 2011) for testing and reporting randomized algorithms. The sample size used for each test was 1000.

$\hat{A}_{12}$: Vargha and Delaney statistics are non-parametric effect size measures. We have a performance measure, $\mathcal{M}$, (fitness function f_i, in Eq. (6)) of two algorithms, $\mathcal{A}$ (SOBS) and $\mathcal{B}$ (PRNG). The value of $\hat{A}_{12}$ is a measure of the probability that algorithm $\mathcal{A}$ yields higher values of $\mathcal{M}$ than algorithm $\mathcal{B}$ does. For example, if $\hat{A}_{12} = 0.8$, then, 80% of the time, algorithm $\mathcal{A}$ will produce better results of $\mathcal{M}$ than algorithm $\mathcal{B}$ will. If $\hat{A}_{12}$ is 0.5, both algorithms are equivalent. Based on this description, the reported $\hat{A}_{12}$ values in Table 4 indicate that in every network size and model, the performance of SOBS is better than that of PRNG. The *effsize* package (Torchiano, 2020) was used for this test.

p-value: This value was calculated using the Mann–Whitney U test, which is a non-parametric test used to determine whether there is a significant difference between the algorithms. Given a performance measure, $\mathcal{M}$, (fitness function f_i) of two algorithms, $\mathcal{A}$ (SOBS) and $\mathcal{B}$ (PRNG), and a significance level, α (we set $\alpha = 0.05$ in this test), if the obtained p-value is less than α, then there is a significant difference in the performance of the algorithms. The p-value obtained from this test is $p = 0.000$ for each network size and model. This means that there is a significant difference in the performance of the algorithms. To implement this test, we used the *mannwhitneyu* method of the *SciPy* (Virtanen et al., 2020) package in Python.

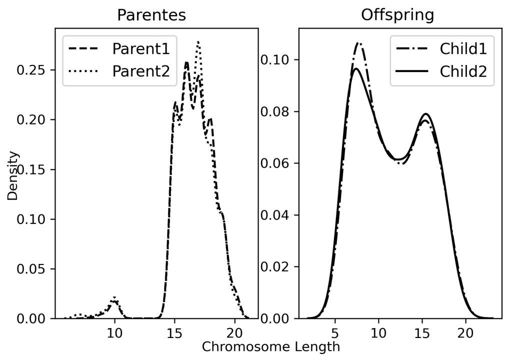
Table 4: Results of $\hat{A}_{12}$ for comparing SOBS and PRNG

Model	**Network Size**							
	16	**24**	**32**	**40**	**48**	**56**	**64**	**72**
WS	0.990	0.998	0.999	0.995	0.999	0.997	0.997	0.999
ER	0.715	0.786	0.998	1.0	0.999	0.996	0.981	1.0
Waxman	0.972	0.995	0.999	1.0	1.0	1.0	1.0	1.0
BA	0.901	0.684	1.0	0.998	0.999	0.999	1.0	0.999

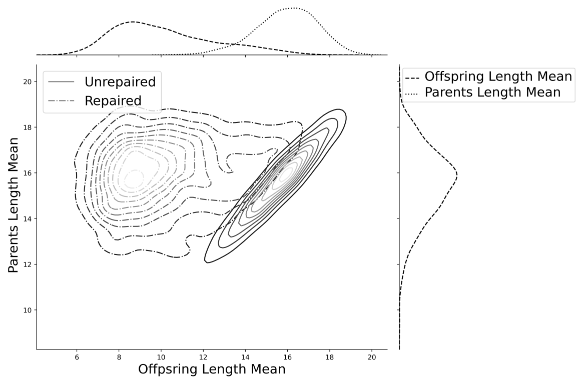
7. Effect of the Repair Function on Solution Lengths

The main objective of the repair function is to repair impaired chromosomes that may appear after the crossover. As designed in (Ahn & Ramakrishna, 2002), the repair function removes the upper part of a chromosome where a loop is discovered and replaces it with a randomly chosen part from the topology. That is, it depends on the topology directly to repair infeasible solutions and obtain other feasible solutions. Without going any further into the design details of the repair function, which is beyond the scope of this work (see Ahn & Ramakrishna, 2002 for details), we justify why Algorithm 1 starts from extreme ends and not from the middle. The following test was designed to clarify the effect of the repair function on the solution length diversity. The typical deterministic network topology shown in Fig. 2 was chosen for this test. The solutions between nodes 1 and 20 were exhausted (301600 solutions obtained); then chromosomes in the middle, namely chromosomes with lengths $\{11, 12, 13, 14\}$, were removed (that is, 69259 removed chromosomes). Then, two chromosomes (parents) were randomly chosen from the remaining 232341 solutions ($301600 - 69259 = 232341$) to be fed to the crossover and repair functions. The lengths of the produced offspring were recorded. This was repeated 1000 times to obtain a sample of 1000.

Fig. 10(a) shows 1000 samples. This indicates that even if chromosomes of lengths $\{11, 12, 13, 14\}$ never occur in the parent chromosomes, those chromosomes repeatedly appear among the offspring. That is, as mentioned previously, the repair function depends directly on the topology to fix the infeasible solutions, and even if some lengths, specifically those in the middle of the initial population lengths set, never occur, the repair function can produce those new lengths. The relationship between the parent chromosome mean length and the offspring mean length when the offspring are repaired or not is shown in the kernel density plot in Fig. 10(b). Data in Fig. 10(b) were obtained as explained earlier, except that no chromosomes were removed before taking a sample of 1000. Fig. 10(b) shows that when a chromosome has not been repaired, the relationship between parent and offspring lengths is perfectly linear.



(a) Parents versus offspring length density.



(b) Relationship between parents and offspring mean length

,

Figure 10: Effect of the repair function on the produced solution lengths during the optimization process

8. Conclusion

In this study, we presented a new initialization algorithm, called SOBS, for a variable-length chromosome GA for the SP. Diversity metrics were provided. The effect of the repair function on solution lengths was also studied.

SOBS provides an initial population with better fitness and balanced genotype diversity and target more phenotype diversity. By using the OBL concept, SOBS provided a *good* initial guess (initial population with better fitness). By using stratification and exploiting the direct representation of the population, SOBS distinguished between solutions with less genotype diversity and those with higher genotype diversity. Stratification also allowed SOBS to achieve more phenotypic diversity. Having higher length-level diversity is preferable because the first characteristic of the optimal solution is its length. The simulation results with a deterministic network and four random-graph models showed that SOBS provides a population with higher fitness than PRNG does and achieves solutions with higher accuracy. According to the Vargha and Delaney statistical tests, the probability that SOBS yields better results is between 68% and 100%. There were only three values that were less than 90%. However, SOBS does not provide a better running time in many cases because SOBS is a *compositional* method and requires more than one step to select the initial population.

The suggested length-based metric (Simpson index) provided an indication of the diversity of initial population in terms of their lengths. The repair function also helped in this type of diversity during the optimization process. However, the effect of the repair function on population diversity needs to be investigated further. Another aspect that deserves further research is the relationship between the graph model and the quality of the initial population produced with SOBS. For instance, small-world networks tend to have higher clustering coefficients and small average shortest path lengths; the relationship of these graph metrics (and perhaps other metrics) with the quality of the initial population when stratification is used deserves further attention.

References

Ahn, C. W., & Ramakrishna, R. S. (2002). A genetic algorithm for shortest path routing problem and the sizing of populations. *IEEE transactions on evolutionary computation*, *6*, 566–579.

Akay, B., & Karaboga, D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information sciences*, *192*, 120–142.

Arcuri, A., & Briand, L. (2011). A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *2011 33rd International Conference on Software Engineering (ICSE)* (pp. 1–10). IEEE.

Ayo, B. S. et al. (2017). An improved genetic algorithm for flight path re-routes with reduced passenger impact. *Journal of Computer and Communications*, *5*, 65.

Balta, E. C., Jain, K., Lin, Y., Tilbury, D., Barton, K., & Mao, Z. M. (2017). Production as a service: A centralized framework for small batch manufacturing. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)* (pp. 382–389). IEEE.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *science*, *286*, 509–512.

Burke, E. K., Gustafson, S., & Kendall, G. (2004). Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, *8*, 47–62.

Chen, M., Wang, K., Dong, X., & Li, H. (2020). Emergency rescue capability evaluation on urban fire stations in china. *Process Safety and Environmental Protection*, *135*, 59–69.

Chen, P., Tong, R., Lu, G., & Wang, Y. (2018). The α-reliable path problem in stochastic road networks with link correlations: A moment-matching-based path finding algorithm. *Expert Systems with Applications*, *110*, 20–32.

Chen, Y., Zeng, X., & Yuan, T. (2017). Design and development of earthquake emergency rescue command system based on gis and gps. In *International Symposium for Intelligent Transportation and Smart City* (pp. 126–138). Springer.

Chua, D. B., Kolaczyk, E. D., & Crovella, M. (2006). Network kriging. *IEEE Journal on Selected Areas in Communications*, *24*, 2263–2272.

Clerc, M. (2008). Initialisations for particle swarm optimisation. *Online at http://clerc. maurice. free. fr/pso*, .

Cong, J., Kahng, A. B., & Leung, K.-S. (1998). Efficient algorithms for the minimum shortest path steiner arborescence problem with applications to vlsi physical design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *17*, 24–39.

De Neve, H., & Van Mieghem, P. (2000). Tamcra: a tunable accuracy multiple constraints routing algorithm. *Computer communications*, *23*, 667–679.

Desaulniers, G., & Soumis, F. (1995). An efficient algorithm to find a shortest path for a car-like robot. *IEEE Transactions on Robotics and Automation*, *11*, 819–828.

Dong, N., Wu, C.-H., Ip, W.-H., Chen, Z.-Q., Chan, C.-Y., & Yung, K.-L. (2012). An opposition-based chaotic ga/pso hybrid algorithm and its application in circle detection. *Computers & Mathematics with Applications*, *64*, 1886–1902.

Dong, O., Yu, P., Liu, H., Feng, L., Li, W., Chen, F., & Shi, L. (2018). A service routing reconstruction approach in cyber-physical power system based on risk balance. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium* (pp. 1–6). IEEE.

Elsayed, S., Sarker, R., & Coello, C. A. C. (2016). Sequence-based deterministic initialization for evolutionary algorithms. *IEEE transactions on cybernetics*, *47*, 2911–2923.

Gama, J., Couceiro, M., Dias, G., & Vaz, V. (2015). Small-world networks in professional football: conceptual model and data. *European Journal of Human Movement*, *35*, 85–113.

Gao, W.-f., Liu, S.-y., & Huang, L.-l. (2012). Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Communications in Nonlinear Science and Numerical Simulation*, *17*, 4316–4327.

Gao, Y., & Wang, Y.-J. (2007). A memetic differential evolutionary algorithm for high dimensional functions' optimization. In *Third International Conference on Natural Computation (ICNC 2007)* (pp. 188–192). IEEE volume 4.

Guo, L., & Matta, I. (2003). Search space reduction in qos routing. *Computer Networks*, *41*, 73–88.

Gupta, A. K., Smith, K. G., & Shalley, C. E. (2006). The interplay between exploration and exploitation. *Academy of management journal*, *49*, 693–706.

Gutiérrez, A., Lanza, M., Barriuso, I., Valle, L., Domingo, M., Perez, J., & Basterrechea, J. (2011). Comparison of different pso initialization techniques for high dimensional search space problems: A test with fss and antenna arrays. In *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)* (pp. 965–969). IEEE.

Hand, D. J. (2010). Statistical analysis of network data: Methods and models by eric d. kolaczyk. *International Statistical Review*, *78*, 135–135.

Helwig, S., & Wanka, R. (2008). Theoretical analysis of initial particle swarm behavior. In *International conference on parallel problem solving from nature* (pp. 889–898). Springer.

Kazimipour, B., Li, X., & Qin, A. K. (2013). Initialization methods for large scale global optimization. In *2013 IEEE Congress on Evolutionary Computation* (pp. 2750–2757). IEEE.

Kazimipour, B., Li, X., & Qin, A. K. (2014). A review of population initialization techniques for evolutionary algorithms. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2585–2592). IEEE.

Kazimipour, B., Salehi, B., & Jahromi, M. Z. (2012). A novel genetic-based instance selection method: Using a divide and conquer approach. In *The 16th*

CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012) (pp. 397–402). IEEE.

Khankhour, H., Abouchabaka, J., & Abdoun, O. (2019). Genetic algorithm for shortest path in ad hoc networks. In *International Conference on Artificial Intelligence and Symbolic Computation* (pp. 145–154). Springer.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (pp. 707–710). volume 10.

Lunacek, M., & Whitley, D. (2006). The dispersion metric and the cma evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 477–484).

Maaranen, H., Miettinen, K., & Mäkelä, M. M. (2004). Quasi-random initial population for genetic algorithms. *Computers & Mathematics with Applications*, *47*, 1885–1895.

Maaranen, H., Miettinen, K., & Penttinen, A. (2007). On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, *37*, 405.

Magurran, A. E. (2013). *Measuring biological diversity*. John Wiley & Sons.

Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K., & Ito, M. (2004). A personal tourism navigation system to support traveling multiple destinations with time restrictions. In *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004.* (pp. 18–21). IEEE volume 2.

Miao, C., Liu, H., Zhu, G. G., & Chen, H. (2018). Connectivity-based optimization of vehicle route and speed for improved fuel economy. *Transportation Research Part C: Emerging Technologies*, *91*, 353–368.

Mohamed, R. E., Saleh, A. I., Abdelrazzak, M., & Samra, A. S. (2018). Survey on wireless sensor network applications and energy efficient routing protocols. *Wireless Personal Communications*, *101*, 1019–1055.

Moy, J. (1994). Open shortest path first version 2. rfq 1583. *Internet Engineering Task Force*, .

Ozer, A. B. (2010). Cide: chaotically initialized differential evolution. *Expert Systems with Applications*, *37*, 4632–4641.

Park, S. K., & Miller, K. W. (1988). Random number generators: good ones are hard to find. *Communications of the ACM*, *31*, 1192–1201.

Qiongbing, Z., & Lixin, D. (2016). A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. *Expert Systems with Applications*, *60*, 183–189.

Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. (2007). Quasi-oppositional differential evolution. In *2007 IEEE congress on evolutionary computation* (pp. 2229–2236). IEEE.

Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. (2008). Opposition versus randomness in soft computing techniques. *Applied Soft Computing*, *8*, 906–918.

Rahnamayan, S., Wang, G. G., & Ventresca, M. (2012). An intuitive distance-based explanation of opposition-based sampling. *Applied Soft Computing*, *12*, 2828–2839.

Rastgoo, R., & Sattari-Naeini, V. (2018). Gsomcr: Multi-constraint genetic-optimized qos-aware routing protocol for smart grids. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, *42*, 185–194.

Rojas-Morales, N., Rojas, M.-C. R., & Ureta, E. M. (2017). A survey and classification of opposition-based metaheuristics. *Computers & Industrial Engineering*, *110*, 424–435.

Rothlauf, F. (2006). Representations for genetic and evolutionary algorithms. In *Representations for Genetic and Evolutionary Algorithms* (pp. 9–32). Springer.

Schuster, H. G., & Just, W. (2006). *Deterministic chaos: an introduction*. John Wiley & Sons.

Senkerik, R., Pluhacek, M., Oplatkova, Z. K., Davendra, D., & Zelinka, I. (2013). Investigation on the differential evolution driven by selected six chaotic systems in the task of reactor geometry optimization. In *2013 IEEE Congress on Evolutionary Computation* (pp. 3087–3094). IEEE.

Shafiee, M. E., & Berglund, E. Z. (2016). Agent-based modeling and evolutionary computation for disseminating public advisories about hazardous material emergencies. *Computers, Environment and Urban Systems*, *57*, 12–25.

Shaikh, A., Rexford, J., & Shin, K. G. (2001). Evaluating the impact of stale link state on quality-of-service routing. *IEEE/ACM Transactions On Networking*, *9*, 162–176.

She, Q., Chen, G., & Chan, R. H. (2016). Evaluating the small-world-ness of a sampled network: Functional connectivity of entorhinal-hippocampal circuitry. *Scientific reports*, *6*, 21468.

Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H., & Laurienti, P. J. (2011). The ubiquity of small-world networks. *Brain connectivity*, *1*, 367–375.

Tiwari, A., & Chang, P.-C. (2015). A block recombination approach to solve green vehicle routing problem. *International Journal of Production Economics*, *164*, 379–387.

Tizhoosh, H. R. (2005). Opposition-based learning: a new scheme for machine intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)* (pp. 695–701). IEEE volume 1.

Torchiano, M. (2020). *effsize: Efficient Effect Size Computation*. URL: `https://CRAN.R-project.org/package=effsize`. doi:`10.5281/zenodo.1480624` r package version 0.8.0.

Uy, N. Q., Hoai, N. X., McKay, R. I., & Tuan, P. M. (2007). Initialising pso with randomised low-discrepancy sequences: the comparative results. In *2007 IEEE Congress on Evolutionary Computation* (pp. 1985–1992). IEEE.

Van Mieghem, P. (2001). Paths in the simple random graph and the waxman graph. *Probability in the Engineering and Informational Sciences*, *15*, 535–555.

Verma, S., Pankaj, R. K., & Leon-Garcia, A. (1998). Qos based multicast routing algorithms for real time applications. *Performance Evaluation*, *34*, 273–294.

Vignesh, V., & Premalatha, K. (2019). Optimal route path sustainability in military information system with reduced interference effect. *The Journal of Supercomputing*, *75*, 6106–6117.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., & Contributors, S. . . (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi:`https://doi.org/10.1038/s41592-019-0686-2`.

Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM (JACM)*, *21*, 168–173.

Wang, X., & Sloan, I. H. (2008). Low discrepancy sequences in high dimensions: How well are their projections distributed? *Journal of Computational and Applied Mathematics*, *213*, 366–386.

WangYuan, & KaiTai, F. (1981). *A note on uniform distribution and experimental design*. Ph.D. thesis.

Zhan, F. B., & Noon, C. E. (1998). Shortest path algorithms: an evaluation using real road networks. *Transportation science*, *32*, 65–73.

Zhao, R.-H., He, W.-Q., Lou, Z.-K., Nie, W.-B., & Ma, X.-Y. (2019). Synchronization optimization of pipeline layout and pipe diameter selection in a self-pressurized drip irrigation network system based on the genetic algorithm. *Water*, *11*, 489.

Zhou, S., Wang, R., Ding, J., Pan, X., Zhou, S., Fang, F., & Zhen, W. (2019). An approach for computing routes without complicated decision points in landmark-based pedestrian navigation. *International Journal of Geographical Information Science*, *33*, 1829–1846.