STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, May 2026, pp 3678–3696.
Published online in International Academic Press (www.IAPress.org)

Parenting Fitness in Genetic Algorithms: Empirical Advantages Over NSGA-II in Logistics Optimization

OUISS Mustapha*, ETTAOUFIK Abdelaziz, MARZAK Abdelaziz

Faculty of Science of Ben Msik, Hassan II University Casablanca, Morocco

Abstract Genetic algorithms (GAs) are population-based metaheuristics widely employed to solve complex real-world problems such as networking and resource allocation. These algorithms evolve a population of candidate solutions through iterative processes of selection, crossover, and mutation. The composition of the next generation is determined by either a general approach, where only offspring are retained, or an elitist approach, which selects fit solutions from the current generation. While elitism enhances solution quality, it is susceptible to premature convergence.
This article presents a comparative study between the parenting fitness mechanism and the crowding distance approach used in the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for solving the Vehicle Routing Problem with Drones (VRPD). Experimental evaluations demonstrate that the proposed parenting fitness method yields consistent improvements in solution quality. Relative improvements range from 27.30% to 43.83% for small problem instances, 6.36% to 11.41% for medium instances, and 0.54% to 1.90% for large instances, with performance variations influenced by the population size. These results validate the effectiveness of parenting fitness as a diversity-preservation strategy in mono-objective optimization.

Keywords Genetic algorithm, nsga-2, optimization, parenting fitness, fitness function, vehicle routing problem with drones

AMS 2010 subject classifications 90B06, 90C47

DOI: 10.19139/soic-2310-5070-2669

1. Introduction

Genetic Algorithms (GAs) are a class of population-based metaheuristics introduced for the first time by [2]. The core mechanics of a Genetic Algorithm involve an iterative process of selection, crossover, mutation and evolution strategy. The algorithm maintains a population of individuals, also called solutions. The encodage of the solutions depends on the problem at hand. These encodings can vary, such as binary strings for discrete problems or real-encoded for continuous problems. Each individual in the population is evaluated by a fitness function. This function, also known as the objective function, measures the quality of the solution it represents. To create a new generation, two individuals are chosen from the current population with a selection mechanism. Common selection mechanisms include roulette wheel selection, tournament selection, or rank-based selection. The genetic material of both selected parents is mixed using a crossover operation to produce new individuals (or offspring). Following crossover, offspring may undergo mutation, where random alterations are made to the genes of the individuals. Choosing the population that will be part of the next generation is governed by the replacement strategy. There are two primary strategies:

- Generational Replacement: The entire parent population is replaced by the offspring population in each iteration.

*Correspondence to: OUISS Mustapha (Email: mustapha.ouiss-etu@etu.univh2c.ma). Faculty of Science of Ben Msik, Hassan II University, Rue Tarik Ibnou Ziad, Casablanca 20360 Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

- Elitist Strategy: Only fit individuals from the current generation, are being kept into the next generation.

The pseudo-code of the elitist genetic algorithm, which is an extension of the simple genetic algorithm (SGA) [3, 4] is illustrated by the algorithm 1.

Algorithm 1 Elitist Genetic Algorithm

Require: Variables initialization : Population size N, maximum generations $G_{\max}$, crossover rate c_r, mutation rate m_r
Ensure: Best solution found

$t \leftarrow 0$
Initialize population $P(t)$
Evaluate fitness of each individual in $P(t)$
while $t < G_{\max}$ **do**
    Create an empty set P_{new}
    while $|P_{\text{new}}| < N$ **do**
        Select two parent individuals p_1, p_2 from $P(t)$ using a selection method
        Generate a uniform random number $r \in [0, 1]$
        if $r < c_r$ **then**
            Apply crossover to p_1 and p_2 to produce offspring c_1, c_2
        end if
        for all offspring $c \in \{c_1, c_2\}$ **do**
            for all gene position i in chromosome of c **do**
                Generate a uniform random number $m \in [0, 1]$
                if $m < m_r$ **then**
                    Apply mutation to the gene at position i in c
                end if
            end for
            Evaluate fitness of c
            Add c to P_{new}
        end for
    end while
    Combine P and P_{new} in P_{all}
    Sort P_{all}
    Select best first N from P_{all} into P_{best}
    $P \leftarrow P_{best}$
    $t \leftarrow t + 1$
end while
return individual with highest fitness in $P(t)$

The elitism used in genetic algorithm can lead to better outcomes. However, elitist Genetic Algorithms are prone to premature convergence, which can result in the loss of valuable genetic diversity (alleles) during the search process. Our aim is to investigate whether the biologically-inspired concept of parenting fitness [1] can offer a competitive alternative for diversity management in mono-objective optimization.

The remainder of this paper is structured as follows: Section II gives a literature review of existing elitist mechanisms. Section III gives an overview of the elitist genetic algorithm with parenting fitness. Section IV presents and discusses the experiments and the computational results, and the last section concludes with future research directions.

2. Literature Review

Algorithms selecting only fit individuals are called elitist algorithms. Each algorithm has a different approach in the way to deal with fit individuals and how they will evolve through generations. In the work of [5], an elitist nondominated sorting genetic algorithm (NSGA II) was presented, which is an evolution of the original NSGA. Each individual of the population had an assigned fitness equal to its nondomination level, from 1 to k, where 1 is the best level, and it decreased until the k level. The algorithm uses the crowding distance which is a diversity preservation mechanism used to maintain a well-distributed set of solutions along the Pareto front. The crowding distance of a solution is computed as the normalized difference between the objective values of its adjacent neighbors. In the work of [6], authors proposed an adaptive EGA which use a greedy algorithm to generate the first population. The elitism in their algorithm is guaranteed by selecting the best p chromosomes of the current population to be part in the next loop. The worst p chromosomes of the current population are deleted to keep the size of the population constant. The modified version of NSGA II called Improved INSGA-II was presented in the article of [7]. The algorithm is quite similar to the NSGA II, with the difference parent seletion, crossover, and mutation strategies. The elitism strategy is performed by selecting the best N individuals from the combined population (parents and offspring). In the work of [8], an elitist genetic algorithm called preservation genetic algorithm (EGA) which preserves the best-found individuals during the execution of the algorithm. these individuals will be replaced by the offspring if they are more fit. The algorithm continues until it reaches a pre-set fitness value or the maximum execution number. In the work of [9], the elitism consists on transferring directly a certain ratio of the best individuals in the current population to the next population. In the work of [10], elitism is applied to preserve the non-redundant previous population, and merge offspring in the population. In the article of [11], 8% of best individuals are copied to the next population without performing crossover and mutation. In the article of [12], authors proposed a Self-Adaptive Simulated Binary Crossover with elitism, that replaces worst individuals by most fit ones. Another incorporation of elitism in the genetic algorithm was presented in the work of [13], that consist on ensuring that new individuals in the current population are fit than the previous iterations. In the work of [14], the best individuals are being selected to be part of the next population even before participating in the crossover and mutation. The table 1 presents a summarization of the studied articles.

Table 1. Literature review summary.

Work	Elitism mechanism
[5]	Non-dominated, and Crowding distance sorting
[6]	Keeping best n individuals
[7]	Keeping best n individuals from the combined population (parents and offspring)
[8]	Preserves the best-found individuals along the execution
[9]	Tranferring a ratio r of the best individuals
[10]	Preserving elitist solutions
[11]	Copying best individuals to the next population without performing crossover and mutation
[12]	Replacing worst individuals by most fit ones
[13]	Ensuring superiority of the current population over the past one
[14]	Copying best individuals to the next population without performing crossover and mutation

These approaches, even if they have different names, use almost the same philosophy by retaining a percentage of best individuals using only the fitness of these solutions. While research is focused exclusively on selecting best individuals according to their fitness, the approach used in this article focuses also on keeping individuals with high generational capacity, using a parameter called parenting fitness which was introduced for the first time in [1]. The primary goal of this study is to provide a proof-of-concept for the parenting fitness mechanism. The NSGA seems to be the most interesting algorithm that uses a particular elitist strategy called crowding distance. NSGA-II is used not as a direct competitor, but as a well-established benchmark whose crowding distance represents a sophisticated, state-of-the-art method for maintaining population diversity.

3. Elitist Genetic Algorithm with Parenting Fitness (eGAwPF)

The parenting fitness parameter introduced for the first time in the work of [1], is a technique that can be implemented into a genetic algorithm to improve the quality of final solutions. Parenting fitness evaluates an individual's ability to produce high-quality offspring, independently to its own fitness score.

3.1. Theoretical Motivation

The strategy of the parenting fitness takes its strength from the theoretical concept of the building blocks [3]. A chromosome can be seen as a composition of building blocks, which are small segments with specific quality. Combining these segments with other segments may lead the algorithm toward optimal solutions by permitting to produce better individual. A good individual is a collection of good segments [3].

Let consider σ a set of potential building blocks ι ($\sigma = \{\iota_0, \iota_1, ..\iota_n\}$) needed to produce the fit individual by the genetic algorithm. The quality ϱ of an individual i is the number of ι in its genotype is as follows:

$$\varrho_i = \sum_{j=0}^{n} \iota_j, \iota \in \sigma \tag{1}$$

Since the building blocks are naturally small [3], the length of each building block ζ should be in the interval $[1, \tau]$ where $\tau \ll L$, where L is the size of an individual. The high value of ϱ_i for the individual i should increase the parenting fitness $\lambda(p)_t$ at generation t. An important point is that in the vehicle routing problem or similar optimization problems, the order of these building blocks is also crucial. Even with good building blocks, wrong ordering may weaken the individual. However, with problems where the ordering is not an issue, they benefit from this high value of ϱ_i.

3.2. Evolution phase in the parenting fitness context

In the context of the genetic algorithm with the parenting fitness, a chromosome with less fit fitness but with a high parenting fitness will be kept alive as long as it is a potential good offspring generator. Using the parenting fitness needs that the algorithm is revisited to consider the parenting fitness parameter in the evolution phase, where the individuals that will be part of the next generation are chosen. The algorithm 2 presents the overall structure of the eGAwPF, and the figure 1 illustrates the philosophy of the parenting fitness integration in the algorithm.

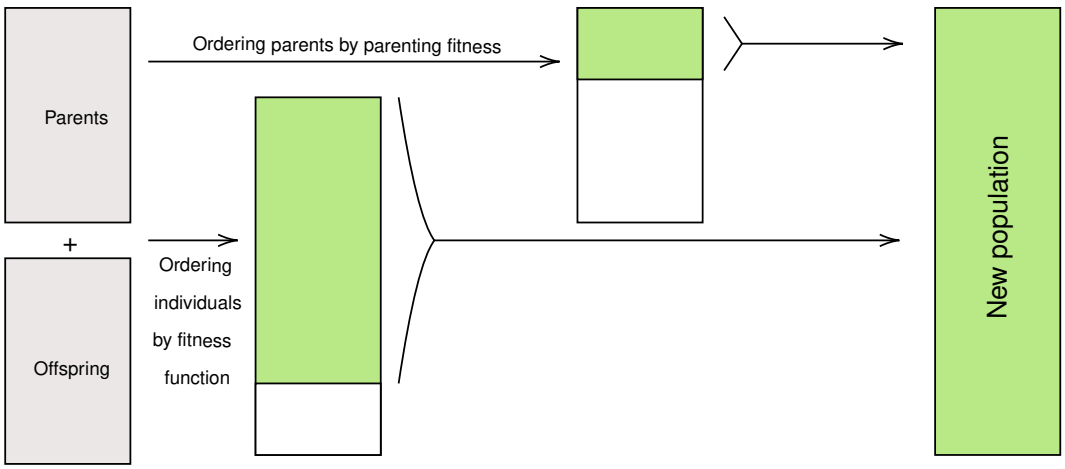

Figure 1. Preparing the next population in the elitist Genetic Algorithm with Parenting Fitness

Algorithm 2 Elitist GA with Parenting Fitness

```
Initialization;
while generation < G_max do
    Evolve the population (selection, crossover, mutation)
    Update parenting fitness of parents
    Prepare next population considering the parenting fitness
end while
```

3.3. Chromosome Structure

Each offspring keeps track of its parents' indices to facilitate the updates. This tracking is possible by using the parent indices (e.g. id_1 and id_2). Also, when implementing the parenting fitness we must add another field (or gene) to the chromosome called pf that will contain the parenting fitness value of the individual. One can use independant array to store the indices and the parenting fitness value. Hopefuly, this addition do not require much memory since the three fields are numerical, hence, does not create memory issues. An integer in python occupes 4 bytes, hence, 3 fields should use 12 bytes multiplies by the number of individuals. Even with a large number of individuals (e.g. 2000), the maximum use of memory is only 24KBs.

3.4. Parenting fitness update

The parenting fitness is used at the last stages of the algorithm's loop, hence, the algorithm's standard operations as the selection, crossover and mutation are not affected by its integration. This ability gives the programmer liberty to choose any approach for these operations. For each offspring created so far, the algorithm calculates the parenting fitness $\lambda(p)_t$ of both parents. In the previous article [1], the value of the parenting fitness was relative to the high fitness value in the whole offspring. However, with this approach, no negative value was assigned to the less fit parents. For this reason, we use in this article the mean value of the offspring population in each loop. With this approach, less fit parents will get a negative notation. The parenting fitness of individuals is calculated by the equation (2).

$$\lambda(p)_t = \sum(\bar{o}_t - f(t)_{op}) \tag{2}$$

where $\bar{o}_t$ is the mean fitness of offspring population of the current generation t, and $f(t)_{op}$ is the fitness function of the offspring o of the selected parent p.

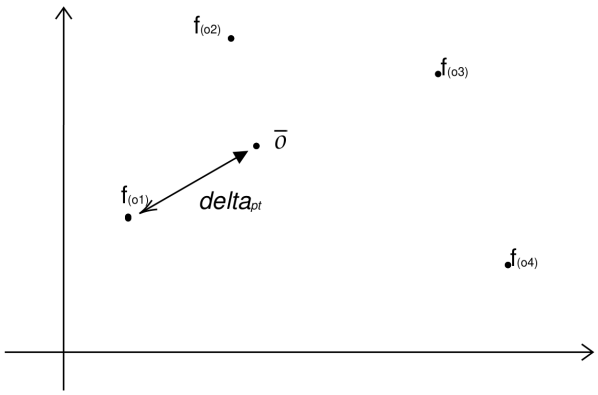

Figure 2. Parenting fitness function representation graph

The figure 2 illustrates visually the calculation function 2. The cumulative excess of the parent's offspring to the mean value defines the strength of the generational ability of the parent.

The parenting fitness of parents is initialized with a predefined value ε as follows:

$$f(n)_{op} = \varepsilon, (\forall n, \varepsilon \in \mathbb{N}) \tag{3}$$

The ε allow to define the begining state of parents, either consider them good offspring generators ($\varepsilon > 0$) or less fit ones ($\varepsilon < 0$).

4. Experiments and Computational results

4.1. Vehicle Routing Problem

The Vehicle Routing Problem, was introduced by Dantzig and Ramser in 1959 [15]. We consider a Vehicle Routing Problem with Drones (VRPD) scenario in which the drone fleet is homogeneous, characterized by identical flight ranges and capacities. Furthermore, the energy consumption per unit distance is considered to be a constant, therefore, this parameter is ignored. The complete mathematical formulation of the VRPD is well detailed in the work of [21]. The objective function in our experiments model is to minimize the maximum distance of all routes L performed by the drone d. The objective function is as follows:

$$obj = min \sum (L_d) \tag{4}$$

The following objective and constraints are considered:

- All points in the dataset must be visited;
- Respect the drone's maximum flight distance: $Ld_i < f_d$;
- Each point must be visited once;
- A central departure point is considered N_0;
- A central arrival point is considered;
- All route begins from N_0 and ends at N_0.

4.2. Crossover

The crossover operator governs the diversification mechanism, enabling the exploration of a broader region within the search space. For the problems as the vehicle routing problems, the Order Crossover (OX) operator [16] is employed in both algorithms. The procedural steps of the OX operator are detailed in Algorithm 3, and a schematic representation is provided in figure 3.

Algorithm 3 Order Crossover (OX) Procedure

1: Randomly select two distinct cut points, i and j (where $i < j$, and $i > 0$), on both parent chromosomes $P_1^{(t)}$ and $P_2^{(t)}$ from generation t.
2: Generate offspring o_1 and o_2 by exchanging the genetic segments between positions i and j from the parents.
3: Remove genes that appear more than once in each offspring.
4: Insert the genes missing in each offspring.

4.3. Mutation

The swap mutation (SM) operator represents one of the natural mutation mechanism for combinatorial problems such as the vehicle routing problem. This operator functions by randomly selecting two genes within a chromosome and exchanging their positions. The procedural steps of the swap mutation operation are formally described in Algorithm 4, with the process further illustrated in Figures 4, 5, and 6.

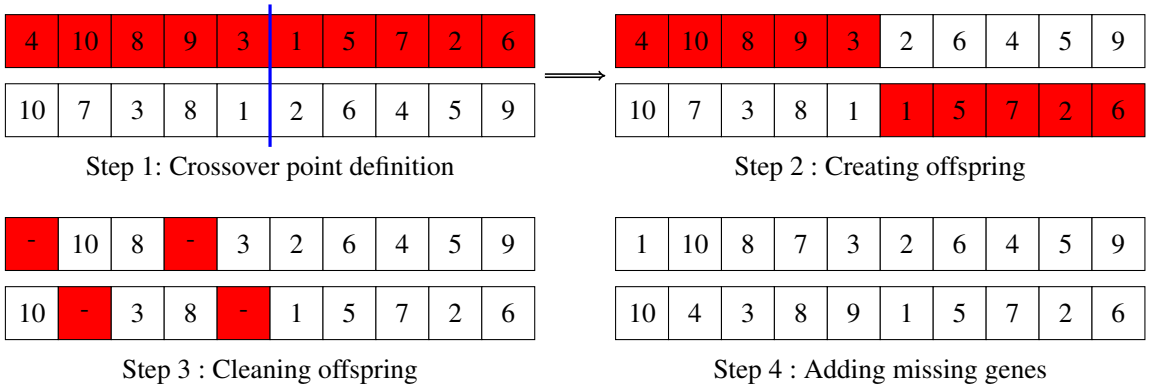

Figure 3. Order Crossover (OX) explanation

Algorithm 4 Swap Mutation Operator

1: Randomly select two distinct positions, p and q, from the chromosome.
2: Swap the alleles at positions p and q.

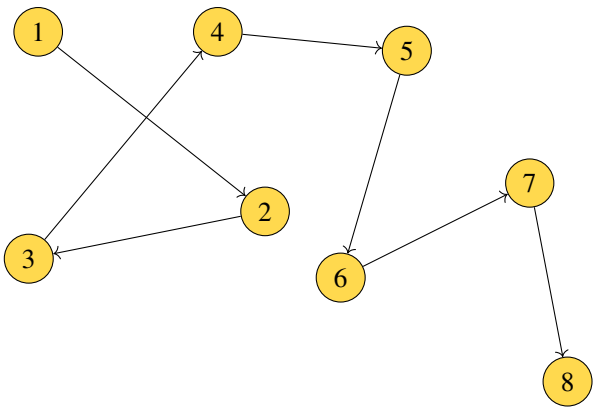

Figure 4. Initial state of a route.

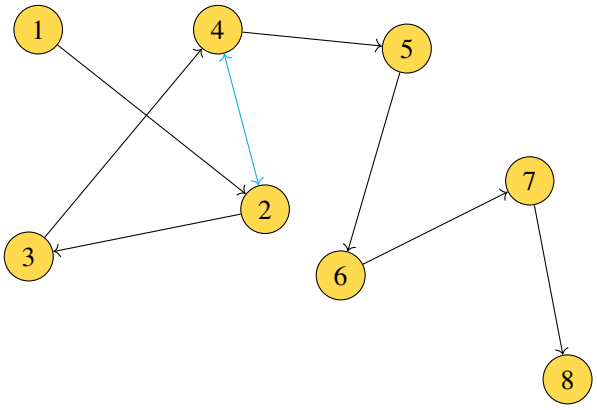

Figure 5. Selecting two positions to swap.

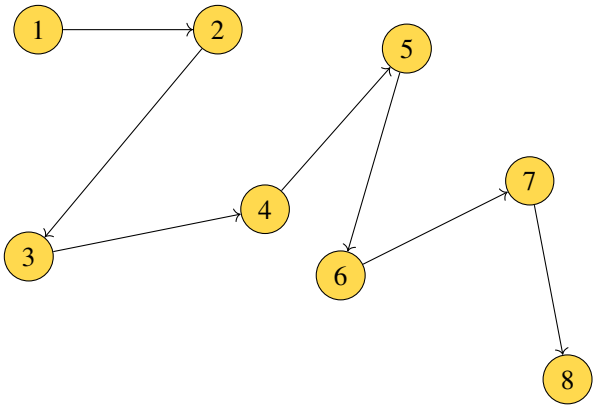

Figure 6. Final result of swap mutation.

4.4. Fitness function

To ensure an equitable comparison between the proposed approach and the established NSGA-II algorithm, an identical fitness function was employed for both. For the VRPD, the fitness function, detailed in Algorithm 5, involves decomposing the chromosome's encoded "big tour" into operationally feasible routes. The DEAP framework [17, 18, 19] was utilized to execute the NSGA-II benchmark, since it is already implemented in the framework and ready to use.

4.5. Dataset description

The algorithms were evaluated using multiple benchmark datasets publicly available at [20]. These specific problem instances were originally introduced and described in the work of [22].

An instance file is designated by the label N.M.T, where N denotes the number of nodes in the dataset, M represents the spatial dimension of the grid, and T is the identifier for the scenario. The file's first line specifies the total nodes in the dataset. Subsequent lines each detail the X-coordinate, Y-coordinate, and demand value for a each node. A separate file contains the depot's coordinates and the specifications of the drone employed. The structure of the dataset file and its corresponding depot file is illustrated in Figure 7.

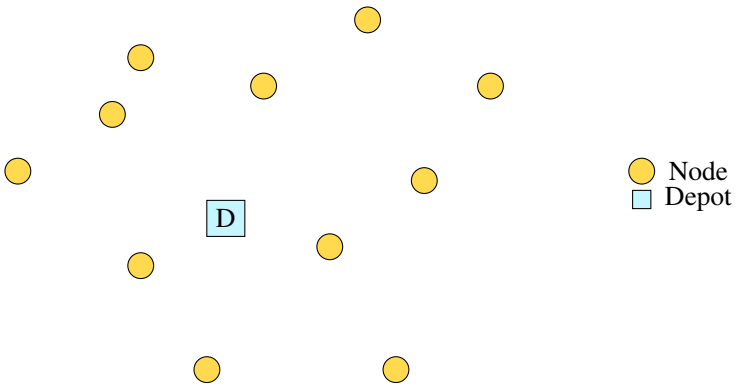

Figure 7. Route Construction : Raw dataset and depot

4.5.1. Route Construction function The figure 8 illustrate the process of constructing a tour that passes through all nodes without considering the constraints linked to the drones, and the figure 9 illustrate the decomposition of the big tour into sub-tours that respects the maximum distance flight of the drone, that is, the algorithm creates a new route each time the constraint is exceeded. The algorithm 5 represents the pseudo-code of the decomposition of the big tour into sub-tours. The fitness of an individual i is the sum of all sub-tours done by the drone. The distance is calculated using the euclidian equation :

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(q_i-p_i\right)^2} \tag{5}$$

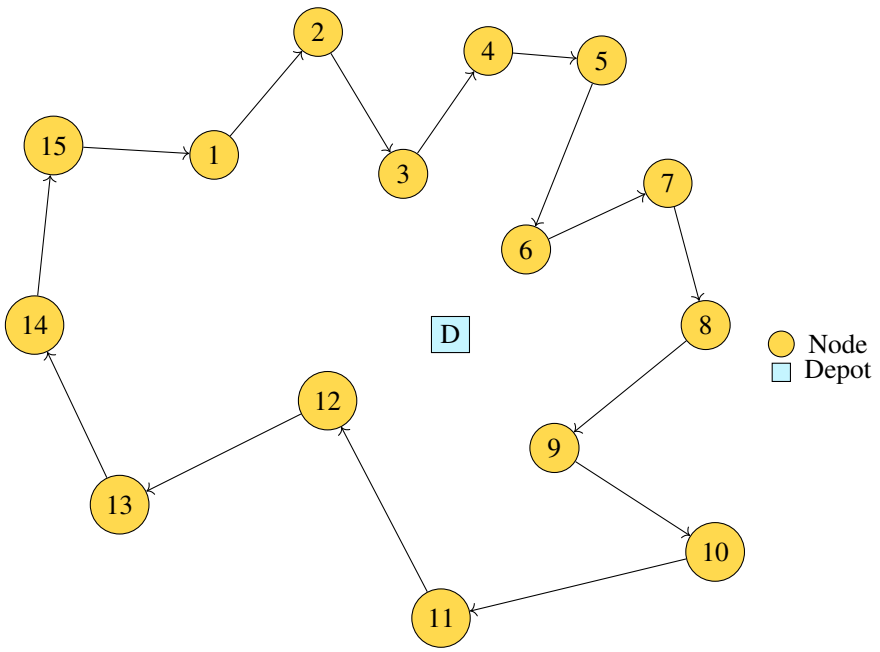

Figure 8. Route Construction : Big tour construction

Algorithm 5 Feasible Route Construction for VRPD

```
function ConstructFeasibleRoutes(chromosome)
    Initialize Maximum Route Length: mrl ← 0
    for all consecutive gene pairs (g_i, g_{i+1}) in chromosome do
        increment the length of the route;
        if The new route exceeds the drone maximum drone flight limit then
            Create new route and add the distance to the total distance;
        else
            Add the distance to the total distance;
        end if
    end for
    return Total distance
end function
```

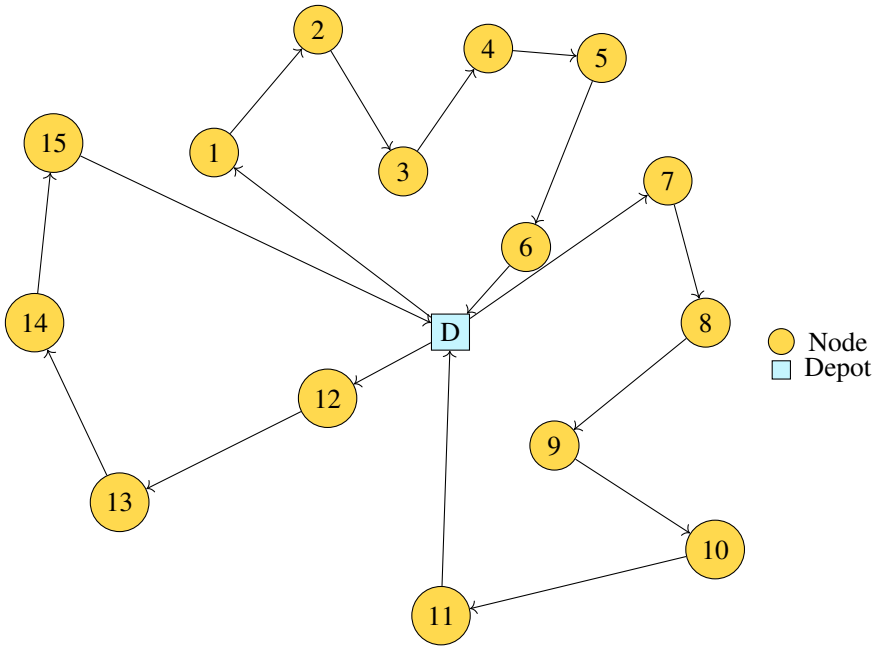

Figure 9. Route Construction : Sub-tours construction

4.5.2. Technical implementation Both genetic algorithms were implemented in Python and executed on an identical hardware configuration: an Intel® i3 dual-core processor (2.20 GHz) with 4 GB of RAM. To ensure a fair comparison, the experiments were conducted under identical stochastic conditions, without fixing a random seed. To explore the wide parameter space, results are presented from a single execution per configuration. The experimental results serve as a proof of concept, demonstrating that the parenting fitness mechanism can yield significant improvements in solution quality in comparison with other techniques, such as crowding distance. This choice prioritizes breadth of exploration over statistical depth.

The shared genetic parameters for the eGAwPF and NSGA-II algorithms, including population size and the number of generations, are detailed in Table 2.

Table 2. Genetic parameters

Parameter	Value
Parent selection	2-Tournament
Crossover operation	order crossover operator (OX)
Crossover rate C_r	0.75
Mutation M_r	0.02

4.5.3. Sensitivity analysis Experiments were conducted with varying population sizes to examine the relationship between the population size and algorithmic performance for both the eGAwPF and NSGA-II. Two configurations were tested:

- **Set 1 (Base configuration):** Population size and generation count equal to the dataset size (N).
- **Set 2 (Expanded configuration):** Population size and generation count equal to twice the dataset size ($2xN$).

Table 3. Computational results of using small population, with the elitist GA with Parenting Fitness (EGAwPF) with continuous update

Instance	NSGA II		EGAwPF (20%)			EGAwPF (10%)			EGAwPF (2%)			Best of eGAwPF	GAP2
	Best	Time(s)	Best	Time(s)	GAP1	Best	Time(s)	GAP1	Best	Time(s)	GAP1		
20.10.3	117,0	0,31	**92,0**	0,74	0,2717	**59,0**	0,62	0,9831	**57,0**	0,63	1,0526	57,0	1,0526
20.10.4	140,0	0,40	**72,0**	0,68	0,9444	**91,0**	0,69	0,5385	**66,0**	0,67	1,1212	66,0	1,1212
20.5.2	42,0	0,21	**34,0**	0,45	0,2353	**33,0**	0,56	0,2727	**32,0**	0,50	0,3125	32,0	0,3125
20.5.3	43,0	0,25	**26,0**	0,56	0,6538	**29,0**	0,44	0,4828	**27,0**	0,46	0,5926	26,0	0,6538
20.5.4	39,0	0,26	**36,0**	0,45	0,0833	**25,0**	0,47	0,5600	**32,0**	0,69	0,2188	25,0	0,5600
50.20.4	858,0	7,15	**745,0**	17,49	0,1517	**704,0**	15,72	0,2188	**644,0**	15,38	0,3323	644,0	0,3323
50.30.1	1173,0	8,23	**1095,0**	16,40	0,0712	**1090,0**	17,49	0,0761	1189,0	16,48	-0,0135	1090,0	0,0761
50.30.2	999,0	7,58	**929,0**	15,84	0,0753	**966,0**	16,21	0,0342	**981,0**	17,46	0,0183	929,0	0,0753
50.30.3	1177,0	8,32	**1128,0**	17,00	0,0434	**1128,0**	16,37	0,0434	**1091,0**	16,15	0,0788	1091,0	0,0788
50.30.4	1170,0	7,64	**1104,0**	16,09	0,0598	**1040,0**	16,04	0,1250	**1003,0**	15,93	0,1665	1003,0	0,1665
200.30.2	4503,0	485,28	**4427,0**	1226,27	0,0172	**4305,0**	1207,91	0,0460	4518,0	1237,90	-0,0033	4305,0	0,0460
200.30.3	4794,0	464,90	**4745,0**	1231,64	0,0103	**4662,0**	1200,91	0,0283	**4737,0**	1222,72	0,0120	4662,0	0,0283
200.30.4	4472,0	464,91	**4417,0**	1244,72	0,0125	**4424,0**	1221,95	0,0108	**4403,0**	1209,06	0,0157	4403,0	0,0157
200.40.1	5894,0	480,66	5954,0	1233,69	-0,0101	**5810,0**	1229,23	0,0145	5898,0	1234,78	-0,0007	5810,0	0,0145
200.40.2	6193,0	467,44	**6169,0**	1218,78	0,0039	**6177,0**	1211,46	0,0026	**6026,0**	1222,79	0,0277	6026,0	0,0277

Table 4. Computational results of using small population, with the elitist GA with Parenting Fitness (EGAwPF) with reinitialization

Instance	NSGA II		EGAwPF (20%)			EGAwPF (10%)			EGAwPF (2%)			Best of eGAwPF	GAP2
	Best	Time(s)	Best	Time(s)	GAP1	Best	Time(s)	GAP1	Best	Time(s)	GAP1		
20.10.3	117,0	0,31	**90,0**	0,64	0,3000	**69,0**	0,67	0,6957	**71,0**	0,59	0,6479	69,0	0,6957
20.10.4	140,0	0,40	**82,0**	0,60	0,7073	**106,0**	0,71	0,3208	**78,0**	0,65	0,7949	78,0	0,7949
20.5.2	42,0	0,21	**40,0**	0,44	0,0500	**37,0**	0,63	0,1351	**39,0**	0,59	0,0769	37,0	0,1351
20.5.3	43,0	0,25	**28,0**	0,48	0,5357	**35,0**	0,54	0,2286	**34,0**	0,45	0,2647	28,0	0,5357
20.5.4	39,0	0,26	**26,0**	0,50	0,5000	**30,0**	0,46	0,3000	**30,0**	0,46	0,3000	26,0	0,5000
50.20.4	858,0	7,15	**700,0**	15,72	0,2257	**795,0**	15,96	0,0792	**781,0**	16,56	0,0986	700,0	0,2257
50.30.1	1173,0	8,23	**1101,0**	16,11	0,0654	**1061,0**	15,50	0,1056	**1039,0**	18,14	0,1290	1039,0	0,1290
50.30.2	999,0	7,58	**986,0**	16,16	0,0132	**993,0**	16,04	0,0060	**920,0**	16,69	0,0859	920,0	0,0859
50.30.3	1177,0	8,32	**1144,0**	16,49	0,0288	**1096,0**	16,88	0,0739	**1155,0**	16,49	0,0190	1096,0	0,0739
50.30.4	1170,0	7,64	**1104,0**	16,24	0,0598	**1063,0**	16,86	0,1007	**1107,0**	16,42	0,0569	1063,0	0,1007
200.30.2	4503,0	485,28	4534,0	1218,65	-0,0068	**4448,0**	1212,97	0,0124	4587,0	1227,77	-0,0183	4448,0	0,0124
200.30.3	4794,0	464,90	**4719,0**	1235,34	0,0159	**4728,0**	1215,48	0,0140	**4735,0**	1241,00	0,0125	4719,0	0,0159
200.30.4	4472,0	464,91	**4373,0**	1213,98	0,0226	**4454,0**	1203,81	0,0040	**4456,0**	1208,13	0,0036	4373,0	0,0226
200.40.1	5894,0	480,66	5914,0	1245,57	-0,0034	5966,0	1230,41	-0,0121	**5865,0**	1256,44	0,0049	5865,0	0,0049
200.40.2	6193,0	467,44	**6174,0**	1234,39	0,0031	**6113,0**	1207,53	0,0131	**6073,0**	1242,69	0,0198	6073,0	0,0198

Table 5. Computational results of using large population, with the elitist GA with Parenting Fitness (EGAwPF) with continuous update

Instance	NSGA II		EGAwPF (20%)			EGAwPF (10%)			EGAwPF (2%)			Best of eGAwPF	GAP2
	Best	Time(s)	Best	Time(s)	GAP1	Best	Time(s)	GAP1	Best	Time(s)	GAP1		
20.10.3	100,0	1,47	**48,0**	2,25	1,0833	**63,0**	2,29	0,5873	**50,0**	2,22	1,0000	48,0	1,0833
20.10.4	112,0	1,30	**66,0**	2,37	0,6970	**60,0**	2,17	0,8667	**54,0**	2,34	1,0741	54,0	1,0741
20.5.2	41,0	0,94	**31,0**	1,76	0,3226	**33,0**	1,69	0,2424	**25,0**	1,68	0,6400	25,0	0,6400
20.5.3	35,0	1,03	**24,0**	1,82	0,4583	**30,0**	1,68	0,1667	**27,0**	1,77	0,2963	24,0	0,4583
20.5.4	37,0	0,99	**25,0**	1,68	0,4800	**28,0**	1,74	0,3214	**30,0**	1,78	0,2333	25,0	0,4800
50.20.4	769,0	26,89	**636,0**	65,11	0,2091	**667,0**	67,40	0,1529	**615,0**	61,76	0,2504	615,0	0,2504
50.30.1	1111,0	29,10	**994,0**	63,70	0,1177	**992,0**	65,98	0,1200	**1018,0**	65,81	0,0914	992,0	0,1200
50.30.2	942,0	31,09	**852,0**	64,12	0,1056	**855,0**	61,35	0,1018	**886,0**	66,92	0,0632	852,0	0,1056
50.30.3	1123,0	29,90	**1033,0**	73,15	0,0871	**1016,0**	67,57	0,1053	**1047,0**	68,59	0,0726	1016,0	0,1053
50.30.4	1129,0	30,61	**980,0**	67,41	0,1520	**1082,0**	67,44	0,0434	**1033,0**	65,48	0,0929	980,0	0,1520
200.30.2	4432,0	1833,04	**4319,0**	4854,14	0,0262	**4234,0**	4833,68	0,0468	**4331,0**	4853,21	0,0233	4234,0	0,0468
200.30.3	4620,0	1847,64	**4537,0**	4902,67	0,0183	**4507,0**	4846,28	0,0251	**4585,0**	4883,10	0,0076	4507,0	0,0251
200.30.4	4255,0	1864,40	**4145,0**	4824,55	0,0265	4330,0	4859,42	-0,0173	**4171,0**	4871,41	0,0201	4145,0	0,0265
200.40.1	5799,0	1930,29	**5745,0**	4887,26	0,0094	**5769,0**	4831,12	0,0052	**5780,0**	4956,99	0,0033	5745,0	0,0094
200.40.2	6152,0	1802,35	**6059,0**	4876,22	0,0153	**5952,0**	4881,07	0,0336	**5911,0**	4810,08	0,0408	5911,0	0,0408

Table 6. Computational results of using large population, with the elitist GA with Parenting Fitness (EGAwPF) with reinitialization

Instance	NSGA II		EGAwPF (20%)			EGAwPF (10%)			EGAwPF (2%)			Best of eGAwPF	GAP2
	Best	Time(s)	Best	Time(s)	GAP1	Best	Time(s)	GAP1	Best	Time(s)	GAP1		
20.10.3	100,0	1,47	**65,0**	2,32	0,5385	**61,0**	2,45	0,6393	**46,0**	2,24	1,1739	46,0	1,1739
20.10.4	112,0	1,30	**68,0**	2,59	0,6471	**67,0**	2,54	0,6716	**66,0**	2,50	0,6970	66,0	0,6970
20.5.2	41,0	0,94	**27,0**	2,06	0,5185	**30,0**	1,96	0,3667	**30,0**	1,93	0,3667	27,0	0,5185
20.5.3	35,0	1,03	**27,0**	1,76	0,2963	**29,0**	1,84	0,2069	**24,0**	1,98	0,4583	24,0	0,4583
20.5.4	37,0	0,99	**27,0**	1,97	0,3704	**26,0**	1,85	0,4231	**28,0**	1,96	0,3214	26,0	0,4231
50.20.4	769,0	26,89	**666,0**	58,56	0,1547	**674,0**	63,05	0,1409	**631,0**	63,48	0,2187	631,0	0,2187
50.30.1	1111,0	29,10	**1084,0**	62,11	0,0249	**1038,0**	66,65	0,0703	1145,0	68,26	-0,0297	1038,0	0,0703
50.30.2	942,0	31,09	**902,0**	66,70	0,0443	**894,0**	72,39	0,0537	**826,0**	66,74	0,1404	826,0	0,1404
50.30.3	1123,0	29,90	**1037,0**	69,50	0,0829	**1005,0**	69,04	0,1174	**990,0**	65,81	0,1343	990,0	0,1343
50.30.4	1129,0	30,61	**1009,0**	70,58	0,1189	**910,0**	68,32	0,2407	**1011,0**	69,33	0,1167	910,0	0,2407
200.30.2	4432,0	1833,04	**4310,0**	4866,12	0,0283	**4244,0**	4833,92	0,0443	4452,0	4707,76	-0,0045	4244,0	0,0443
200.30.3	4620,0	1847,64	**4557,0**	4908,59	0,0138	**4620,0**	4886,16	0,0000	4654,0	4600,88	-0,0073	4557,0	0,0138
200.30.4	**4255,0**	1864,40	4430,0	4855,47	-0,0395	4314,0	4821,24	-0,0137	4339,0	4543,27	-0,0194	4314,0	-0,0137
200.40.1	5799,0	1930,29	**5725,0**	4874,57	0,0129	5848,0	4885,56	-0,0084	**5672,0**	4635,12	0,0224	5725,0	0,0129
200.40.2	6152,0	1802,35	**6031,0**	4874,73	0,0201	**6061,0**	4873,32	0,0150	6221,0	4636,85	-0,0111	6031,0	0,0201

4.5.4. Parenting fitness variation To investigate the effect of parenting fitness persistence, two variants of the algorithm improved with the parenting fitness parameter were implemented: one with continuous update of the parenting fitness value of each survival parent, and the other where this value is reinitialized to ε. The continuous update allow ancestors (individuals existing in generation from and before $t-1$) to exist in order to understand the power of building blocks over generations.

Furthermore, the algorithm was executed with the parental selection rate configured to 2, 10%, and 20% of the best parents in each population. We tested small (2%), moderate (10%), and large (20%) retention rates to explore the spectrum from weak to strong elitism based on parenting fitness. The remainder of the subsequent generation is populated by the best individuals selected from the combined pool of offspring and the current population. This multi-rate analysis provides insights into the influence of the parental contribution proportion on the algorithm's performance and convergence characteristics. In the tables, values in bold indicate where the eGAwPF variant found a better solution than the NSGA-II baseline for that specific instance.

The presented results are supplemented by the calculation of two performance gaps, using the following formulas:

$$Gap(1) = \frac{BV(N) - BV(E)}{BV(E)} \tag{6}$$

$$Gap(2) = \frac{BV(N) - BV^*(E)}{BV^*(E)} \tag{7}$$

where $BV(E)$ denotes the best fitness value achieved by the eGAwPF, and $BV(N)$ represents the corresponding best fitness value obtained by the NSGA-II algorithm. The value $BV^*(E)$ is defined as the optimal fitness discovered by the eGAwPF across its three tested parental fitness percentages.

4.6. Execution analysis

The core genetic algorithm comprises four primary operators: parent selection, crossover, mutation, and fitness evaluation, each exhibiting a computational complexity of O(n). As the algorithm executes for n generations, its overall complexity is $O(n^2)$. The introduced parenting fitness function also operates in O(n) time, as it iterates over the population to update the parental fitness values. Consequently, the enhanced algorithm (eGAwPF) demonstrates a longer execution time compared to the standard NSGA-II. This performance difference is attributable to three additional computational steps required in each generation:

- **Parenting Fitness Update:** Needs an $O(n)$ computational step per generation.
- **Dual Sorting Operations:** The evolutionary phase requires sorting the population twice: first by parenting fitness, and by solution fitness.

The last two steps depends on the programmation language sorting implementation.

4.7. Relative Improvement

Let $\overline{N}$ and $\overline{E}$ represent the mean result values obtained by NSGA-II and the eGAwPF, respectively, across the small, medium, and large problem instances. The relative improvement is calculated using Equation (8), and it quantifies the percentage reduction in fitness (distance) achieved by eGAwPF compared to NSGA-II. The corresponding results are presented in Tables 7, 8, 9, and 10.

$$\Delta = \frac{\overline{N} - \overline{E}}{\overline{N}} \times 100\% \tag{8}$$

Table 7. Relative improvement (large population, continuous update of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	65,00	38,80	42,80	37,20
	median	41,00	31,00	33,00	30,00
	relative improvement (%)		**40,31**	**34,15**	**42,77**
medium instances	mean	1014,80	899,00	922,40	919,80
	median	1111,00	980,00	992,00	1018,00
	relative improvement (%)		**11,41**	**9,11**	**9,36**
large instances	mean	5051,60	4961,00	4958,40	4955,60
	median	4620,00	4537,00	4507,00	4585,00
	relative improvement (%)		**1,79**	**1,84**	**1,90**

Table 8. Relative improvement (large population, with reinitialisation of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	65,00	42,80	42,60	38,80
	median	41,00	27,00	30,00	30,00
	relative improvement (%)		**34,15**	**34,46**	**40,31**
medium instances	mean	1014,80	939,60	904,20	920,60
	median	1111,00	1009,00	910,00	990,00
	relative improvement (%)		**7,41**	**10,90**	**9,28**
large instances	mean	5051,60	5010,60	5017,40	5067,60
	median	4620,00	4557,00	4620,00	4654,00
	relative improvement (%)		**0,81**	**0,68**	-0,32

Table 9. Relative improvement (small population, continuous update of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	76,20	52,00	47,40	42,80
	median	43,00	36,00	33,00	32,00
	relative improvement (%)		**31,76**	**37,80**	**43,83**
medium instances	mean	1075,40	1000,20	985,60	981,60
	median	1170,00	1095,00	1040,00	1003,00
	relative improvement (%)		**6,99**	**8,35**	**8,72**
large instances	mean	5171,20	5142,40	5075,60	5116,40
	median	4794,00	4745,00	4662,00	4737,00
	relative improvement (%)		**0,56**	**1,85**	**1,06**

4.8. Execution Time analysis

Let τ be the time penalty of the execution of the eGAwPF in comparison with NSGA-II. The used equation to calculate the time penalty is defined by the equation (9), where T^{E} and T^{N} are the mean execution time for the eGAwPF and NSGA respectively. Tables 11,12,13,14 represent the calculation of the time penalty.

$$\tau = \frac{T^{\mathrm{E}} - T^{\mathrm{N}}}{T^{\mathrm{N}}} \times 100\% \tag{9}$$

Table 10. Relative improvement (small population, with reinitialisation of parenting fitness)

		NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	76,20	53,20	55,40	50,40
	median	43,00	40,00	37,00	39,00
	relative improvement (%)		**30,18**	**27,30**	**33,86**
medium instances	mean	1075,40	1007,00	1001,60	1000,40
	median	1170,00	1101,00	1061,00	1039,00
	relative improvement (%)		**6,36**	**6,86**	**6,97**
large instances	mean	5171,20	5142,80	5141,80	5143,20
	median	4794,00	4719,00	4728,00	4735,00
	relative improvement (%)		**0,55**	**0,57**	**0,54**

Table 11. Time penalty (large population, continuous update of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	1,15	1,97	1,91	1,96
	median	1,03	1,82	1,74	1,78
	time penalty (%)		72,39	67,14	70,72
medium instances	mean	29,52	66,70	65,95	65,71
	median	29,90	65,11	67,40	65,81
	time penalty (%)		125,97	123,43	122,63
large instances	mean	1855,54	4868,97	4850,31	4874,96
	median	1847,64	4876,22	4846,28	4871,41
	time penalty (%)		162,40	161,40	162,72

Table 12. Time penalty (large population, with reinitialisation of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	1,15	2,14	2,13	2,12
	median	1,03	2,06	1,96	1,98
	time penalty (%)		86,91	85,91	85,15
medium instances	mean	29,52	65,49	67,89	66,72
	median	29,90	66,70	68,32	66,74
	time penalty (%)		121,88	130,01	126,06
large instances	mean	1855,54	4875,90	4860,04	4624,78
	median	1847,64	4874,57	4873,32	4635,12
	time penalty (%)		162,77	161,92	149,24

Table 13. Time penalty (small population, continuous update of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	0,29	0,58	0,56	0,59
	median	0,26	0,56	0,56	0,63
	time penalty (%)		**101,47**	**94,83**	**105,39**
medium instances	mean	7,78	16,56	16,37	16,28
	median	7,64	16,40	16,21	16,15
	time penalty (%)		**112,81**	**110,26**	**109,18**
large instances	mean	472,64	1231,02	1214,29	1225,45
	median	467,44	1231,64	1211,46	1222,79
	time penalty (%)		**160,46**	**156,92**	**159,28**

Table 14. Time penalty (small population, with reinitialisation of parenting fitness)

	Instance	NSGA II	EGAwPF (20%)	EGAwPF (10%)	EGAwPF (2%)
small instances	mean	0,29	0,53	0,60	0,55
	median	0,26	0,50	0,63	0,59
	time penalty (%)		86,09	109,77	91,89
medium instances	mean	7,78	16,14	16,25	16,86
	median	7,64	16,16	16,04	16,56
	time penalty (%)		107,38	108,74	116,64
large instances	mean	472,64	1229,58	1214,04	1235,21
	median	467,44	1234,39	1212,97	1241,00
	time penalty (%)		160,15	156,86	161,34

5. Discussion and Conclusions

Based on the experimental results, the genetic algorithm using the parenting fitness parameter finds better solutions than the NSGA-II algorithm and its crowding distance across both experimental configurations. When utilizing a small population size, the relative improvement ranges from 27.30% to 43.83% for small problem instances, 6.36% to 8.72% for medium instances, and 0.54% to 1.85% for large instances. Employing a larger population yielded further improvements, with relative gains of 34.15% to 42.77% for small instances, 7.41% to 11.41% for medium instances, and 0.68% to 1.90% for large instances. For large instances, it may need additional execution loops, since the search space can be significantly vast. Also, these results substantiate the hypothesis that the retention of high-fitness parents across generations may significantly enhance algorithmic performance. This is visible when comparing the results and the gap from experiments of both continuous and reinitialisation approach.

Limitations

It is important to note that while the eGAwPF excels in the mono-objective context of the VRPD, where the goal is to minimize the maximum route length, it is not currently designed for multi-objective optimization, a domain where NSGA-II is particularly robust. Additionally, the eGAwPF incurs a higher mean execution time compared to NSGA-II. This computational overhead will be addressed in future work through the implementation of parallel processing techniques.

Future research will focus on the following points:

- implementing a master-slave parallelization to delegate the parenting fitness update to the slave processes, which may significantly reduce the execution time.

- adaptive parenting fitness rate will be considered to balance the pressure on choosing good parents over generations.

Acknowledgement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors

REFERENCES

1. M. Ouiss, A. Ettaoufik, A. Marzak, and A. Tragha, *Genetic algorithm parenting fitness*, Math. Model. Comput., vol. 10, no. 2, pp. 566–574, 2023.
2. J. H. Holland, *Adaptation in Natural and Artificial Systems*, The MIT Press, 2nd ed, 1992 (first edition 1975 in University of Michigan Press).
3. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, inc, 1989.
4. D. M. D. Vose, *The Simple Genetic Algorithm: Foundations and Theory*, Complex Adapt. Syst., vol. Complex ad, 1999.
5. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, 2002.
6. Y. Zhu, K. Y. Lee, and Y. Wang, *Adaptive Elitist Genetic Algorithm with Improved Neighbor Routing Initialization for Electric Vehicle Routing Problems*, IEEE Access, vol. 9, 2021.
7. Z. Wang, Y. Li, K. Shuai, W. Zhu, B. Chen, and K. Chen, *Multi-objective Trajectory Planning Method based on the Improved Elitist Non-dominated Sorting Genetic Algorithm*, Chinese J. Mech. Eng. (English Ed.), vol. 35, no. 1, 2022.
8. L. Jiang et al., *Temperature prediction of battery energy storage plant based on EGA-BiLSTM*, Energy Reports, vol. 8, 2022.
9. F. Yaman and A. E. Yilmaz, *Elitist Genetic Algorithm Performance on the Uniform Circular Antenna Array Pattern Synthesis Problem*, Przegląd Elektrotechniczny, vol. 88, no. 1b, 2012.
10. S. Rani, B. Suri, and R. Goyal, *On the effectiveness of using elitist genetic algorithm in mutation testing*, Symmetry (Basel), vol. 11, no. 9, 2019.
11. B. V. Natesha and R. M. R. Guddeti, *Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment*, J. Netw. Comput. Appl., vol. 178, 2021.
12. A. Fanggidae, M. I. C. Prasetyo, Y. T. Polly, and M. Boru, *New Approach of Self-Adaptive Simulated Binary Crossover-Elitism in Genetic Algorithms for Numerical Function Optimization*, Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 14s, 2024.
13. H. Jiang, W. Zeng, W. Wei, and X. Tan, *A bilevel flight collaborative scheduling model with traffic scenario adaptation: An arrival prior perspective*, Comput. Oper. Res., vol. 161, 2024.
14. Y. Feng, A. (Avi) Ceder, S. Zhang, and Z. Cao, *Bus routing fine-tuning for integrated network-based demand and bus bridging for a disrupted railway system*, Expert Syst. Appl., vol. 242, 2024.
15. G. B. Dantzig and J. H. Ramser, *The Truck Dispatching Problem*, Manage. Sci., vol. 6, no. 1, pp. 80–91, 1959.
16. P. Moscato, *On genetic crossover operators for relative order preservation*, C3P Rep., 1989.
17. F.-A. Fortin, U. Marc-André Gardner, M. Parizeau, and C. Gagné, *DEAP: Evolutionary Algorithms Made Easy*, 2012.
18. DEAP: Distributed Evolutionary Algorithms in Python. https://pypi.org/project/deap/, 2024.
19. J. Kim and S. Yoo, *Software review: DEAP (Distributed Evolutionary Algorithm in Python) library*, Genet. Program. Evolvable Mach., vol. 20, no. 1, pp. 139–142, 2019.
20. D. Sacramento, https://zenodo.org/records/1403150, 2019.
21. C. C. Murray and A. G. Chu, *The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery*, Transp. Res. Part C Emerg. Technol., vol. 54, pp. 86–109, 2015.
22. D. Sacramento, D. Pisinger, and S. Ropke, *An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones*, Transp. Res. Part C Emerg. Technol., vol. 102, pp. 289–315, 2019.