MATHEMATICAL MODELING AND COMPUTING, Vol. 10, No. 2, pp. 566–574 (2023)

Genetic algorithm parenting fitness

Ouiss M., Ettaoufik A., Marzak A., Tragha A.

Faculty of Sciences Ben M'Sik, Hassan II University of Casablanca, Casablanca, Morocco

(Received 29 January 2023; Accepted 10 April 2023)

The evolution scheme phase, in which the genetic algorithms select individuals that will form the new population, had an important impact on these algorithms. Many approaches exist in the literature. However, these approaches consider only the value of the fitness function to differenciate best solutions from the worst ones. This article introduces the parenting fitness, a novel parameter, that defines the capacity of an individual to produce fittest offsprings. Combining the standard fitness function and the parenting fitness helps the genetic algorithm to be more efficient, hence, producing best results.

Keywords: *genetic algorithm; fitness function; parenting fitness; optimization.*

2010 MSC: 90B06, 90C47 **DOI:** 10.23939/mmc2023.02.566

1. Introduction

For genetic algorithms, selecting individuals that will still alive for the next generations is an important part of the algorithm. This phase is called the evolution phase. There are many strategies to choose these individuals: (1) general, and (2) elitist strategy. The general approach is a straightforward approach that replace the whole current population by the offsprings generated so far. In this strategy, parents and offsprings compete for survival. On the other hand, with the elitist approach, a portion of the best individuals, with high fitness value (or low fitness value, depending on the problem) are selected from the current loop. The remaining part is completed with most fit offsprings. The selection rate is defined in percentage or in number.

In this article, we introduced a novel parameter called parenting fitness (PF). PF defines the capacity of an individual to produce fittest individuals, independently of it fitness function value.

The article is structured as follows: the first section will introduce the genetic algorithm. The second section will discuss the parenting fitness parameter. The vehicle routing problem (VRP) with drones will be discussed in the third section, as a study to test the efficiency of the parenting fitness parameter. The last section will discuss results of experiences, and gives some developing points for future research.

2. Genetic algorithm

Genetic algorithms (GAs) were introduced for the first time by [1]. They may be considered as a special case of the Random Heuristic Search (RHS) [2]. GAs are called population-based metaheuristic, since they manipulate populations of individuals (also called solutions, chromosomes). They aim to find good solutions to complex and time consuming problems. The individuals evolve thru generations according to the mechanism of selection, and genetic operations as crossover and mutation. Genetic algorithms are constructed from a number of reusable components: (1) the chromosome encoding, (2) the fitness function, (3) the selection method, (4) the recombination (crossover and mutation), and (5) the replacement strategy. This type of construction is considered as one of the strengths of GAs, since we can reuse them, or change them without breaking the algorithm.

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics [3]. The basic form of Genetic Algorithms is called simple genetic algorithm (SGA), and it

was well detailed in [3] and [2]. Genetic algorithms are iterated until the fitness value stabilizes or the maximum iteration number is reached. The GAs use the survival mechanism that let fittest individuals to pass thru generations. In each generation, survived individuals mate with other survivals to produce new solutions. These newly produced individuals may be weak or best according to their fitness value. The calculation of the fitness function depends on the problem to be solved. It may be a minimization or maximization function, of a combination of both.

The simple genetic algorithm can be detailed as follows.

Algorithm 1 Simple GA pseudo-code

```
1: initialization: t := 0;
2: initialization crossover rate: random_χ;
3: initialization mutation rate: random_μ;
4: Create initial population P(0);
5: while t ≠ generation_number
6:    Select parent P1(t) and P2(t) from current population;
7:    Generate random_number between 0 and 1
8:    if random_χ > random_number then
9:       Perform crossover χ selected parents;
10:      for all offspring ∈ newly_created_offsprings
11:         for all gene ∈ Genes
12:            Generate random_number between 0 and 1;
13:            if random_μ > random_number then
14:               Perform mutation μ on gene;
15:   Prepare next population;
```

Where χ defines the crossover operator. The crossover is responsible of mixing the genetic informations of the selected parents to produce new offsprings. The mutation operation μ, for example flips alleles from the value 0 to 1 and vice versa, with a certain uniform probability. Crossover and mutation will be discussed in the next sections.

2.1. Chromosome encoding

Genetic algorithms manipulate a population of chromosomes. Each chromosome contains N genes. These genes represent the genotype of the chromosome. The representation of chromosomes is problem dependent, they may be set as bit-string, string, float, or whatever presentation. In addition, when programming with Oriented Object Programming (OOP) we may define genes as classes, that contain complex data for each gene; for example a chromosome can be a representation of a string of genes representing a coordinate point each. The bit-string representation is the commonly used. For example, the OneMax problem uses a string of bits, each bit has a value of 0 or 1, and the goal is to reach a solution with the maximum values of 1's. In the knapsack problem, each gene represents a number between a maximum and minimum, and the goal is to reach a solution with the maximum total of genes values that fill completely the knapsack. In certain cases, the chromosomes may have a variable number of genes, this called a variable-length chromosome. In this case, the standard crossover is no longer applicable of these types of chromosomes.

2.2. Fitness function

The fitness function is a computation that evaluates the quality of the chromosome depending on the current problem to be solve. The fitness computation will for example measure how fit each potential solution is. Fittest solutions may have a maximized value, or minimized value. However, the calculation of the fitness function can take a tremendous amount of time. The implementation of the fitness

function is a critical operation, and the practitioner must take a particular care when implementing it. For simple problem as OneMax problem, knapsack problem, the implementation is straightforward. However, for complex problem with multiple objectives and constraints, the implementation of the fitness function must be done carefully, in order to avoid miscalculations. A beautiful definition was given by [4]: 'The fitness function is described as "fitness function is the only chance that you have to communicate your intentions to the powerful process that genetic programming represents. Make sure that it communicates precisely what you desire".'

2.3. Selection schemes

The objective of selection is to choose the most fit individuals in the current population that will create offsprings by the process of mating. The mating does not guarantee that the offsprings produced are most fit than their parents. The selection mechanism vary according to the scheme used. These schemes can be grouped in four main categories [5]: (1) proportionate reproduction, (2) ranking selection, (3) tournament selection, and (4) steady state selection. We may find in the literature other variantes and other schemes.

2.4. Crossover

The genetic material of two selected parents is mixing using the crossover operation. This operation allows creating new individuals (also called offsprings). This operation may occur with a certain probability defined before the execution of the algorithm. There are many types of crossover in the literature as:

— One-point crossover: a crossover point between 0 and n is chosen with uniform probability.
— Multi-point crossover: a sequence of crossover points is chosen along the chromosome length interchanging at each crossover.
— The order crossover [6]: a randomly selected substring of two parent strings is swapped, creating two new offsprings. The existing genes in the offsprings are deleted, and their positions filled with remaining values.
— Uniform crossover [7]: a string of bits of the chromosomes size is generated with uniform probability. For each 1-bit in the string (or mask), in the n position is swapped. The uniform crossover can be considered as a special case of the multi-point crossover.

We may note that these types of crossover do not perform well for each problem. For example, the order crossover is more suited for vehicle routing problems. The works of [7] demonstrates that one-point crossover performs well than two-points for certain problems, and the uniform crossover is the better choice for others. The crossover operation for variable length chromosome is generally different from fixed length chromosome.

2.5. Mutation

Genetic algorithms perform the mutation next after the crossover operation. Mutation helps GAs to prevent premature convergence and to improve found solutions. In this stage, a random number in the interval $[0, 1]$ is generated with uniform probability for each allele of the chromosome, and these allele values are changed if the randomly generated number is less than the mutation rate.

Many mutation types exist as:

— Bit Flip Mutation: select one or more random bits and flip them.
— Random Resetting: extension of the bit flip for the integer representation. In this case, a random value from the set of permissible values is assigned to a randomly chosen gene.
— Swap Mutation: select two positions on the chromosome at random, and interchange the values.
— Scramble Mutation: from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.

— Inversion Mutation: selects a subset of genes like in scramble mutation, but instead of shuffling the subset, merely inverts the entire string in the subset.

Yet, not every type is suitable for every problem. For example, the swap mutation type is more suitable for sequence problems as vehicle routing problems.

2.6. Evolution strategy

In the evolution phase, genetic algorithms select individuals that will form the next population. There are several evolutionary schemes [8–10] that can be used. Generally, there are three families: (1) the complete replacement: where the whole population is replaced with the offsprings generated; (2) the elitist strategies: where only the best individuals (or a portion) from the current generation are selected; and (3) the steady state strategies: the offsprings are inserted in the current population, and participate in the mating (parent selection). Generally, the number of the offsprings is equal to the number of parents, since the algorithm is executed till the number of offsprings reach the number of the population size.

Complete replacement strategy. With this strategy, the whole population is replaced with the created offsprings. The offsprings are now the current parents. The genetic algorithm is ready to perform another loop. This strategy is simple, no need of sorting neither the current population, nor the offsprings. However, important individuals and alleles can be (and will be) lost from generation to another. The work of [11] brings more details of alleles loss. The premature convergence is more likely to happen with this strategy.

Ellitist strategies. These strategies select only the best individuals from the current generation. The rate is defined in percentage or in number, and the rest is filled with most fit offsprings. Another approach is that the parents and offsprings compete for survival. In this case, the current population and the offsprings are sorted by their fitness, and the N first solutions (or the X percentage) are selected to pass to the next generation. The remaining places are filled by the remaining offsprings. We may find an alternative approach. After the combination of the current population and the offsprings, the first half of the combination is selected to pass to the next population. Best chromosomes have chance to live cross many generations. Its true for the fittest chromosomes. The selection rate must be selected wisely. Too low rate can produce weak final solution, and too high rate converges somehow the ellitist strategy to a local minima.

Steady state strategy. In this strategy, after the mating, and the genetic operations (crossover and mutation), the offsprings may replace the weakest individuals in the current population. The newly created offsprings become parents, and compete for mating. With this strategy, the generations overlap, hence, no need for sorting population.

The choice of evolutionary scheme is an important aspect of GA design and will depend on the nature of the solution space being searched. Yet, the widely used scheme is replacement-with-elitism [10].

3. Parenting fitness

Selecting only fittest individuals can cause premature convergence to a local optimum. We can ask ourselves a question: can a weak individual produce fittest descendants? In order to respond to this question, we introduced a novel parameter called 'Parenting fitness' or 'Parenthood fitness'. The idea behind the parenting fitness parameter is the capacity of an individual to produce, from mating, fittest offsprings despite on how it is the fitness value of the concerned individual. This mechanism can be explained by the fact that an individual can have in its genotype a portion (or multiple portions) that, combined with other portions, can converge to fittest solutions. The parenting fitness defines the capacity of the chromosome to generate fittest individuals. Individual with high parenting fitness may have weak personal fitness (the fitness function). In this case, the evolution phase takes in consideration this parameter, and keeps individuals with high parenting fitness thru generations.

4. Vehicle routing problem

In order to test the efficiency of the parenting fitness, we consider the well known Vehicle Routing Problem (VRP). The VRP was introduced for the first time by [12] as "Truck Dispatching Problem". They presented the problem of routing a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal.

The VRP is considered as a generalization of the well known Traveling Salesman Problem (TSP). The traditional version of VRP considers one depot, and the vehicles capacity as the only constraints, and assume that all vehicle are homogeneous.

The major difference between TSP and VRP is that in the TSP, the salesman (or the carrier) can serve every point on one trip. Instead, in the VRP, the vehicle can only make a limited number of deliveries on each trip. The sum of the quantity delivered can not exceed the vehicle's capacity.

In the VRP, the goal is to serve delivery points, and for each points N_i deliver a quantity q_i. Constraints and objectives may be considered as minimizing vehicle mileage, transportation cost, time window, and so on.

The classical VRP known as Capacitated VRP (CVRP), and other VRP variants can be solved using exact algorithms in order to find the optimal solution. The drawback of these algorithms is the execution time. They can be solved using heuristics, or metaheuristics. However, there are no guarantee that we will find the optimal solution in the end since (meta-)heuristics are stochastic methods.

An overview of exact and approximate algorithms to solve the CVRP is given by [13] as: (1) The assignment lower bound and a related branch-and-bound algorithm, (2) The k-degree center tree and a related algorithm, (3) Dynamic programming, (4) The Clarke and Wright heuristic, (5) The sweep algorithm, among others.

A mathematical formulation for the classical VRP was dressed in the work of [13] as follows.

Let consider $G = (N, A)$ as a graph where $N = 1, \ldots, n$ is a set of vertices representing cities with a central depot at vertex 1, and A is the set of arcs. Each arc(i, j) with $i \neq j$ is associated with a numerical value. The interpretation of this value depends on the context. It can be interpreted as travel cost or travel time, or both in some cases. The distance matrix C can be either symmetrical or asymmetrical, depending on the context. The CVRP consider an unique central depot, and $N_+ = N\backslash\{0\}$ as the set of nodes to be visited. We also consider a set M of available homogeneous vehicles with a capacity D. In other variants of VRP as HVRP, vehicles can have different capacities. The resolution of a CVRP considers the following constraints: (1) each city identified by a vertex is visited only once, by a unique vehicle; (2) each vehicle has a limited capacity; and (3) each vehicle start the route from the central depot, and must return to it in the end.

In the real world VRP problem, the constraints are way more complex than the canonical CVRP constraints. These constaints define the VRP variants. Table 1 gives a quick overview of some existing variants.

Table 1. Definitions.

MDVRP	multi depot VRP, in which we consider more than one central depot
VRPTW	VRP with time window constraints
HVRP	heterogeneous VRP, in which we consider vehicles with different capacity
VRPD or UAVRP	VRP with drones, or Unmanned Aerial VRP. This variant consider drones

VRP with drones is also known as unmanned aerial VRP (UAVRP). UAVRP are the VRP variant that considers drones in the conception of routes. Vehicle routing problems with drones is an extension of the Capacitated Vehicle routing problems. VRPD have raised an enormous interest in the last decade. In the first years of the century, the majority of articles published in relation with drones are focused on military fields. Recent articles are more likely redirected to consider civil application domains, like delivery, transportation, healthcare, surveillance, logistics and so on.

Recent articles are more focused on Delivery problems, especially the last-mile delivery since drones are known to have a limited capacity and a limited battery. Because of these limitations, drone routing must be optimized.

Solving approaches for the VRPD can be exact solutions, heuristics, metaheuristics, or a combination of two or more approaches. A large number of VRPD problems are formulated and resolved using Mixed Integer Linear Programming (MILP).

4.1. Mathematical formulation

We consider a VRPD where an unique drone with a defined maximum flight range. In this initial article, we consider the energy consumption by mile to be constant. The instances used in our test are taken from [14]. These instances contain X and Y coordinates, and a demande value for each customer. The demande value are ignored since we consider drone to perform monitoring missions, which means that demand value is useless. The maximum load capacity of the drone is not considered too. The routes are performed in a 2D environnement. The drone is a multi-rotor type, which means that it can perform multi-directional flights.

The VRPD mathematical formulation is quite similar to the FSTSP of [15], with additional constraints, and some difference for others. We refer to the previously article in the construction of the current model. The objective function of this model is to minimize the maximum distance of all routes performed be the drone. Minimizing the number of routes is not consider in our case.

Table 2 presents the considered notations.

Table 2. Notations.

$N = 1, \ldots, n$	set of nodes representing the points to visit.
N_0	the departure node, and also the final arrival node.
N_+	the set of nodes to which the drone may visit.
$L_{i,j}$	Distance between node i and j.
R	defines all routes performed by the drone.
d	maximum flight range of the drone.

The formulation used in our model is as follows:

$$\min \sum_{i=1}^{N} R_i, \tag{1}$$

subject to the constraints:

$$R_i \leqslant d, \tag{2}$$

$$\sum_{j \in N_0} x_{ij} + \sum_{j \in N_0} \sum_{k \in N_+} y_{ijk} = 1, \quad i \neq j, \quad (i,j,k) \in P, \quad \forall j \in C, \tag{3}$$

$$\sum_{j \in N_+} x_{0j} = 1, \tag{4}$$

$$\sum_{i \in N_0} x_{i,c+1} = 1, \tag{5}$$

$$\sum_{j \in N_0} x_{i,j} = \sum_{k \in N_+} x_{j,k}, \quad \forall j \in C \tag{6}$$

The objective function 1 seeks to minimize the total distance of all routes performed by the drone. The constraint 2 guarantees that the length of a route R_i should be less or equal to the maximum flight distance d of the drone. Constraint 3 requires each node to be visited exactly once. Constraint 4 ensures that the drone takes off from the departure node exactly once, while the constraint 5 requires the drone to return to the landing node exactly once. We consider the take off node is the same as the landing node. Constraint 6 indicates that the drone visiting a node j must also depart from j. Since we ignore the maximum drone load, no constraint is defined for this parameter. In addition, we consider an infinite battery change.

4.2. Genetic algorithm with parenting fitness

The algorithm implements two evolution strategies: (1) Parent and Offspring Replacement Strategy, and (2) Parent and Offspring Replacement with Parenthood Fitness Strategy. The parenting fitness of individuals selected for mating is updated in the end of the loop. For each offsprings created so far, their two parents are being updated. The update is relative to fitness of the offspring in comparison with the maximum fitness of the whole offsprings. The parenting value of individuals may increase or decrease depending on the Maximum fitness of the current population. In the evolution phase, individuals parents are sorted by their parenting fitness. The first n (with $n < Pop\_size$) parents are selected to form the future population. A second sorting is performed according to the fitness value of individuals (parents and offsprings), and the algorithm selects the remaining $Pop\_size - n$ individuals. A new population is ready to loop. This evolution scheme belongs to elitist strategies.

The datasets are composed of multiple files with 20, 50, and 200 visiting points. The algorithm implements two evolution strategies: (1) Parent and Offspring Replacement Strategy, and (2) Parent and Offspring Replacement with Parenthood Fitness Strategy. We ran the algorithm for each set of files.

Table 3 presents the considered parameters.

Table 3. Genetic algorithm parameters.

Parent selection	Roulette Wheel Selection defined as $p_i(t) = \frac{f(a_i(t))}{\sum_{l=1}^{n} f(a_j(t))}$
Fitness function	Maxmimum route distance
Crossover operation	order crossover operator (OX)
Crossover rate C_r	0.75
Mutation M_r	0.02
Evolution scheme	Elitist
Parenting fitness	10% of the population (parents only)

An individual (or chromosome) represents the large tour that a drone takes to visit nodes clustered to the departure node N_0. We consider the order crossover operator (OX) [6], and the swap mutation operator. The crossover controls the diversification mechanism, and the mutation controls the intensification mechanism. The order crossover operates as follows: after the random selection of parents from the mating pool, a randomly selected substring of two parent strings are swapped, creating two new offsprings. Since the offspring should have different nodes, the nodes included in the substring added will be deleted, and their positions will be filled with remaining ones. In swap mutation, the genetic algorithm interchanges the values of two positions randomly selected on the chromosome.

4.3. Experiences and results

The genetic algorithm was implemented using the Python language, and executed on a Intel®i3 machine, with double core 2.20GHz, and 4 Gb RAM. The datasets consist on two files, the first contains informations about departure points, drones that will be launched from these points, and for each drone, the maximum flight range in km. The second dataset contains coordinates of nodes to be visited. In this first article, we will consider only one departure point.

Table 4. Genetic algorithm parameters.

number of generations	$N \times 10$
number of individuals	size of $N \times 2$

The datasets are composed of multiple files with 20, 50, and 200 visiting points. The algorithm implements two evolution strategies: (1) Parent and Offspring Replacement Strategy, and (2) Parent and Offspring Replacement with Parenthood Fitness Strategy.

Table 4 presents the considered parameters.

The number of generation is fixed at 250 for datasets of 200 nodes, to avoid the tremendous execution time. In our future articles, we will consider an HPC environnement to deal with the calculation load. We executed the algorithm several times, and took the best results from these executions.

The results are shown in Tables 5–7.

Table 5. Experiences for dataset of 20 nodes.

Dataset	*Pop_size*	Loop	ES	Max fitness	ES	Max fitness
20.10.3	40	200	PAOR	**46.0**	PAORwPF	**46.0**
20.10.4	40	200	PAOR	57.0	PAORwPF	**50.0**
20.5.2	40	200	PAOR	**26.0**	PAORwPF	**26.0**
20.5.3	40	200	PAOR	24.0	PAORwPF	**23.0**
20.5.4	40	200	PAOR	**24.0**	PAORwPF	25.0
20.10.3	10	200	PAOR	**67.0**	PAORwPF	83.0
20.10.4	10	200	PAOR	**68.0**	PAORwPF	75.0
20.5.2	10	200	PAOR	**32.0**	PAORwPF	36.0
20.5.3	10	200	PAOR	31.0	PAORwPF	**30.0**
20.5.4	10	200	PAOR	**23.0**	PAORwPF	31.0

ES = Evolution strategy. PAOR = Parent and Offspring Replacement Strategy.
Pop_size = Population Size. PAORwPF = PAO with Parenthood Fitness Strategy.

Table 6. Experiences for dataset of 50 nodes.

Dataset	*Pop_size*	Loop	ES	Max fitness	ES	Max fitness
50.20.4	100	500	PAOR	596.0	PAORwPF	**500.0**
50.30.1	100	500	PAOR	990.0	PAORwPF	**954.0**
50.30.2	100	500	PAOR	**801.0**	PAORwPF	813.0
50.30.3	100	500	PAOR	**958.0**	PAORwPF	993.0
50.30.4	100	500	PAOR	**926.0**	PAORwPF	942.0
50.20.4	25	500	PAOR	716.0	PAORwPF	**693.0**
50.30.1	25	500	PAOR	1084.0	PAORwPF	**1075.0**
50.30.2	25	500	PAOR	936.0	PAORwPF	**918.0**
50.30.3	25	500	PAOR	**1032.0**	PAORwPF	1088.0
50.30.4	25	500	PAOR	1036.0	PAORwPF	**1012.0**

ES = Evolution strategy. PAOR = Parent and Offspring Replacement Strategy.
Pop_size = Population Size. PAORwPF = PAO with Parenthood Fitness Strategy.

Table 7. Experiences for dataset of 200 nodes.

Dataset	*Pop_size*	Loop	ES	Max fitness	ES	Max fitness
200.30.2	400	250	PAOR	**4197.0**	PAORwPF	4441.0
200.30.3	400	250	PAOR	4591.0	PAORwPF	**4451.0**
200.30.4	400	250	PAOR	4357.0	PAORwPF	**4163.0**
200.40.1	400	250	PAOR	5815.0	PAORwPF	**5715.0**
200.40.2	400	250	PAOR	5958.0	PAORwPF	**5953.0**

ES = Evolution strategy. PAOR = Parent and Offspring Replacement Strategy.
Pop_size = Population Size. PAORwPF = PAO with Parenthood Fitness Strategy.

From these results, we notice that the parenting fitness give almost the same performance for small dataset (with 20 points) for both replacement strategies. However, using the parenting fitness gives good results for large datasets (with 200 nodes). In addition, a small population size gives also good results. This point is interesting since the genetic algorithm will consume less time to give acceptable solutions. The parenting fitness parameter seems to be a promising enhancement for genetic algorithms. The challenge is to find the optimal function to calculate its value efficiently.

5. Conclusions and future work

The use of the parenting fitness in the genetic algorithm help finding way better individuals in comparison with algorithm using only the fitness function as criterion of best individuals. The main reason is that the parents with portion of potential best fitness pass thru the generation even if they have weak fitness value as individual. In addition, using large population improves the final results. However, raising the population size above a certain value does not give better solutions, but raises only the

time execution of the algorithm. This is due to the random mechanism of genes generation, that may generate redundant individuals.

In future work, we will consider using hybridization with other metaheuristics in order to enhance the parenting fitness function. In addition, the execution of the algorithm will be done in an HPC environnement, using GPUs.

[1] Holland J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press (1975).

[2] Vose M. D. The Simple Genetic Algorithm: Foundations and Theory. Complex Adaptive Systems. MIT Press (1999).

[3] Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc. (1989).

[4] Kenneth E., Kinnear Jr. Advances in Genetic Programming. MIT Press (1994).

[5] Goldberg D. E., Kalyanmoy D. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. Foundations of Genetic Algorithms. **1**, 69–93 (1991).

[6] Moscato P. On genetic crossover operators for relative order preservation. C3P Report (1989).

[7] Syswerda G. Uniform Crossover in Genetic Algorithms. Proceedings of the 3rd International Conference on Genetic Algorithms. 2–9 (1989).

[8] Saini N. Review of Selection Methods in Genetic Algorithms. International Journal of Engineering and Computer Science. **6** (12), 22261–22263 (2017).

[9] Dianati M., Song I., Treiber M. An Introduction to Genetic Algorithms and Evolution Strategies (2002).

[10] McCall J. Genetic algorithms for modelling and optimisation. Journal of Computational and Applied Mathematics. **184** (1), 205–222 (2005).

[11] De J K. A. An Analysis of the Behavior of a Class of Genetic Adaptive Systems (1975).

[12] Dantzig G. B., Ramser J. H. The Truck Dispatching Problem. Management Science. **6** (1), 80–91 (1959).

[13] Laporte G. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research. **59** (3), 345–358 (1992).

[14] Sacramento D. Vehicle Routing Problem with Drones Instances. `https://zenodo.org/record/1403150`.

[15] Murray C. C., Chu A. G. The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-assisted Parcel Delivery. Transportation Research Part C: Emerging Technologies. **54**, 86–109 (2015).

Генетичний алгоритм виховання

Уіс М., Еттауфік А., Марзак А., Трага А.

Факультет наук Бен М'Сік, Університет Хасана II Касабланки, Касабланка, Марокко

Фазова схема еволюції, в якій генетичні алгоритми відбирають особин, що сформують нову популяцію, мала важливий вплив на ці алгоритми. У літературі існує багато підходів. Однак ці підходи враховують лише значення функції відповідності, щоб відрізнити найкращі рішення від гірших. Ця стаття знайомить із придатністю для батьківства, новим параметром, який визначає здатність індивіда народжувати найпридатніших нащадків. Поєднання стандартної фітнес–функції та батьківської придатності допомагає генетичному алгоритму бути ефективнішим і, отже, досягати найкращих результатів.