Introdução à Meta-heurística

Genetic Algorithms

Sandro Carvalho Izidoro

Universidade Federal de Itajubá - UNIFEI
Campus Itabira

May 24, 2022

Summary

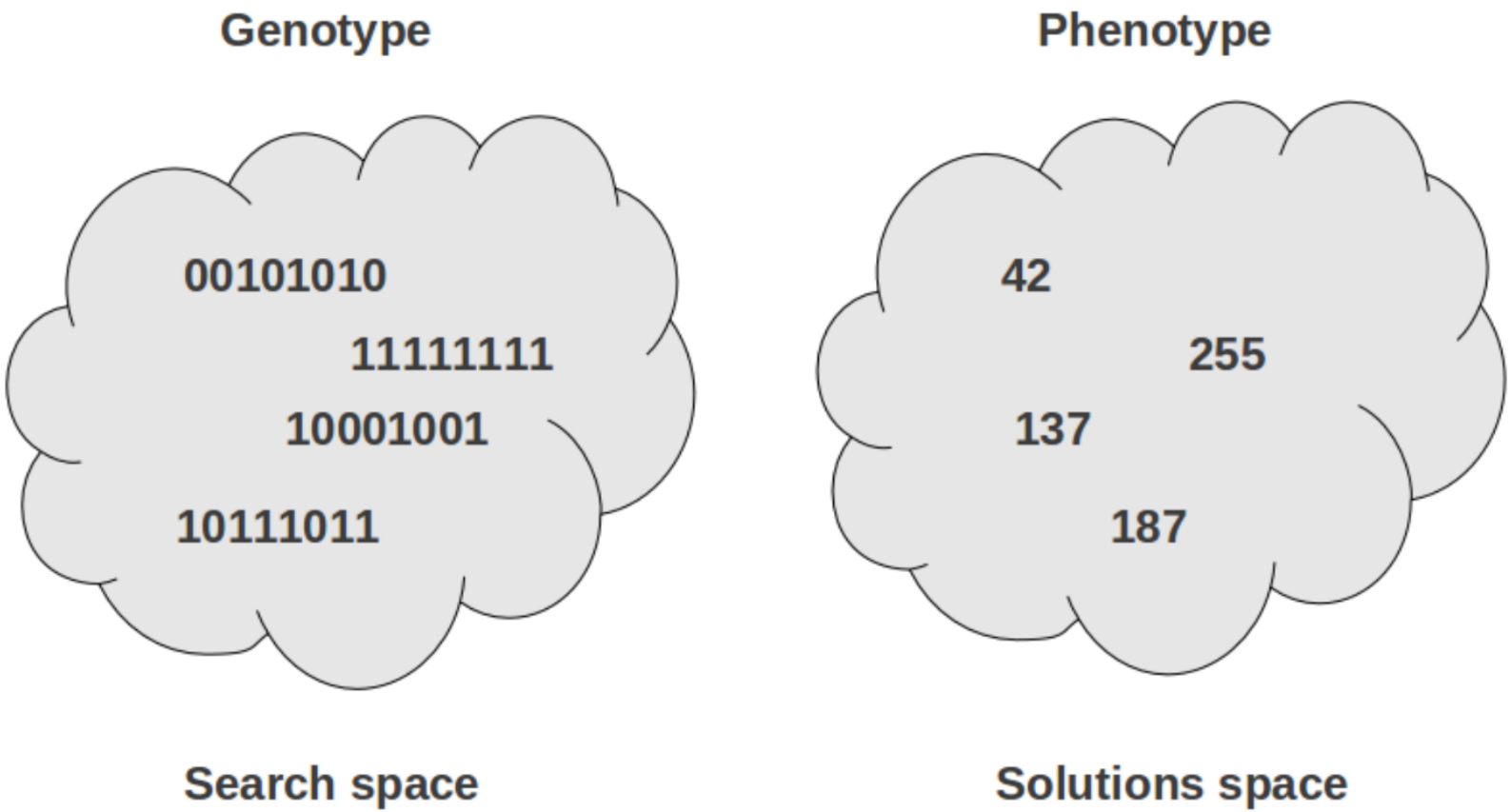
Introduction to Genetic Algorithms

Introduction

- Most dissimilar technique in EAs.
- Introduced by Holland in 1972, and developed by one of its students, Goldberg.
- Individuals are binary strings.
- Chromosome (individual) fixed size.
- There is a mapping of the genotype to the phenotype.

Introduction

Genotype x Phenotype



Introduction

- Operators are applied to the genotype.
- The problem space is known as the search space and encompasses all possible solutions to a given problem.

Operation mode

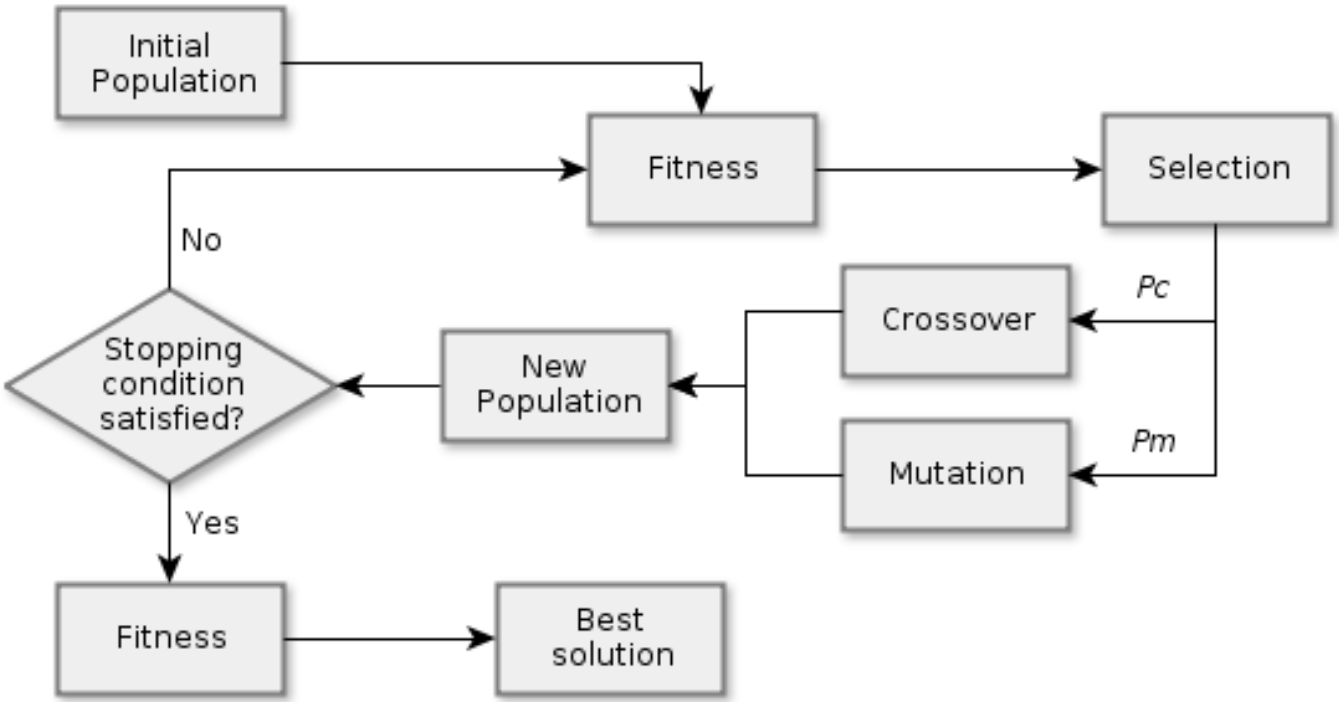
- **Generational:** Every population is replaced by a new population with each generation (traditional GA).
- **Steady State:** Only one or two individuals are produced with each new generation.

Introduction

- Genetic algorithms (GAs) are metaheuristic methods based on Charles Darwin's theory of natural selection and were initially proposed by J. H. Holland in 1972 (Katoch et al., 2020).
- GAs are iterative procedures that evolve a population of individuals, where each individual represents a candidate solution to the problem in question. At each iteration, called generation, the best individuals are selected based on a fitness function. Genetic operators (crossing and mutation) are applied to selected individuals, aiming to produce new individuals from their parents' genetic material.
- This process is repeated until a stop condition is satisfied, which can be a defined number of generations, a detection of convergence or the time of execution of the GA (Izidoro et al., 2014; Eiben and Smith, 2007).

Introduction

Genetic Algorithm



Components of a Genetic Algorithm

Representation of the individual

- The representation of an individual corresponds to the first stage of modelling a GA. An individual from an GA is an abstraction from an individual from the real world. Defining an individual involves simplifying aspects of the real world and represents a possible solution to the problem in question. The modelling of individuals must be carried out in such a way that they can be evaluated, selected and manipulated by genetic operators, and are generally defined by specialists in the field (Eiben and Smith, 2007).

- An individual, or candidate solution, can be represented according to some models described by Eiben and Smith (2007): **binary**, **integer**, **real** and **permutation**.

- The **binary** representation is the simplest, where an individual consists of a simple binary string of digits. The length of the string will depend on the context of the problem and how the individual will be mapped from the real world to the GA individual. A problem with this approach is that different bits have different meanings and a simple change in one of the bits (for example, through a mutation) can bring about very different results.

Components of a Genetic Algorithm

Representation of the individual

- The representation of the **integer** type is a way of defining individuals of an GA when the problem naturally maps different genes (characteristics of an individual) into an element of a set.
- Another way to represent individuals in an GA is through **real** or **floating-point** values. This form consists of using real numbers to compose the string and is used to represent genes with continuous values, and no longer discrete as in the representation of the **integer** type. It is useful to describe, for example, values of distances, heights or weights.
- The **permutation** representation is useful for problems involving sorting, such as ordering tasks or optimization problems (for example, the travelling salesman problem). In this representation, each individual is formed by a string of numbers that represent the sequence for solving the problem.

Components of a Genetic Algorithm

Population

- A set of individuals forms the population of an GA. This population contains possible solutions to the problem and can be generated randomly or through **seeds**.
- Diaz-Gomez and Hougen (2007) cite several factors to be taken into account when defining a randomly generated initial population: the search space, fitness function, diversity, difficulty of the problem, selection and the number of individuals.
- Populations initiated through **seeds** are usually created from pre-processing done with random individuals, where they are evaluated according to the fitness function. Those best evaluated (best score) will compose the initial population (Meadows et al., 2013).

Components of a Genetic Algorithm

Evaluation function (Fitness)

- An evaluation function or fitness function must be able to represent the requirements that a population must adapt to advance to the next generation (Eiben and Smith, 2007). All individuals in the population of each generation of GA are evaluated for this function.
- It is important that the fitness function is representative and can accurately differentiate good (bad) individuals (solutions). An unadjusted fitness function in the assessment of individuals may end up discarding a promising individual, who could help to find better solutions to the problem, in addition to the fact that it consumes resources in individuals who add little to the development of GA.
- Multi-objective fitness functions take different aspects of the problem into account when evaluating individuals, and can treat them equally or give different weights to each one.

Components of a Genetic Algorithm

Selection methods

- The selection operation implies how the choice of individuals to form descendants for the next generation should be made (Mitchell, 1998). The term selective pressure is often used to show how much a selection method considers the evaluation value of individuals (Camargo, 2006). The objective of a selection method is to highlight individuals more apt in the population so that they can generate even better descendants.

- Eiben and Smith (2007) cite several ways to carry out the selection of individuals in an GA, such as the **roulette method**, the selection based on the **absolute value of fitness**, the selection by **tournament** and the selection by **ranking**.

- The **roulette method** is a common way of selecting individuals based on their expected fitness value. A roulette area is provided for each individual, where the size of that area is proportional to the individual's fitness value. The roulette wheel is then spun N times, where N is the number of individuals in the population, and at the end of each time the roulette wheel is spun, the marked individual is selected to be a parent of the next generation (Mitchell, 1998).

Components of a Genetic Algorithm

Selection methods

- The principle of selection by **fitness** is based on the fact that individuals are selected only according to the absolute value at which the fitness function evaluates them. The fittest individuals tend to occupy the entire population very quickly, making the search process more focused on a region of the specific search space. This makes it more difficult for the GA to cover all possible solutions to the problem. This phenomenon is known as premature convergence (Eiben and Smith, 2007).

- Selection by the **tournament** method causes an N number of random individuals to be selected. Then, a random number R is chosen between 0 and 1. If $R < K$ (where K is a parameter, such as 0.75) the fittest individual is selected, otherwise the least fit (Mitchell, 1998). The higher the K value, the greater the selective pressure imposed on the population.

- The selection made through the **ranking** method classifies individuals based on their fitness and then selection probabilities are allocated according to the ranking (and not with the fitness value) (Eiben and Smith, 2007). This approach prevents most of the selection from being made by more able individuals, reducing the selective pressure. It is an alternative to avoid premature convergence (Mitchell, 1998).

Components of a Genetic Algorithm

Genetic operators

- After selecting individuals, two genetic operators are used to generate a new population (next generation) of GA: **crossover** and **mutation**. These genetic operators aim to refine and spread the search, respectively, also bringing more genetic variability.
- The **crossover** operator uses the combination of two individuals (defined here as parents) to generate descendant individuals (defined as children) (Dréo et al., 2006).
- Basically, the crossover operation takes place when two individuals are selected and random parts of them are exchanged between them, thus forming new individuals. Three main ways of crossing can be mentioned: **simple point** (or one-point), **multipoint** (or k-points) and **uniform**.

Components of a Genetic Algorithm

Genetic operators

- The **single point** type crossover selects two parent individuals for the crossing and randomly selects a *Pi* point in these individuals (where $i >= 0$ and $i < n$, being n the size of the individual) and then two child individuals are created by combining the parts of the parent individuals that were divided by the *Pi* point (Umbarkar, 2015).

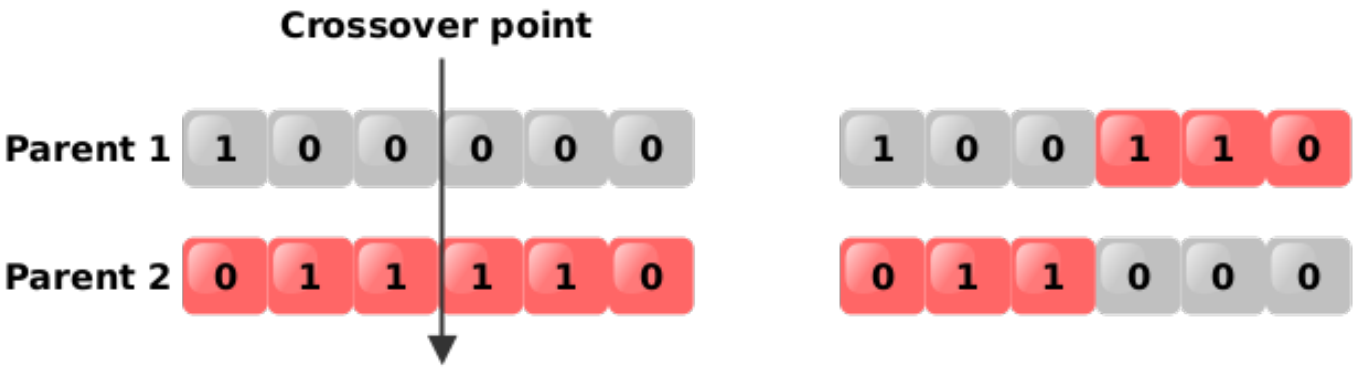


Figure: One-point crossover operation.

Components of a Genetic Algorithm

Genetic operators

- The **multipoint** crossing acts very similarly to the single point crossover, however, in this case, more than one point is created. The method selects two parent individuals and also randomly selects a value of K, which determines the points $P1i$ to $Pk-1i$ (where $i >= 0$ and $i < n$, being n the size of the individual) that will be the places where the crossover will take place (Umbarkar , 2015).

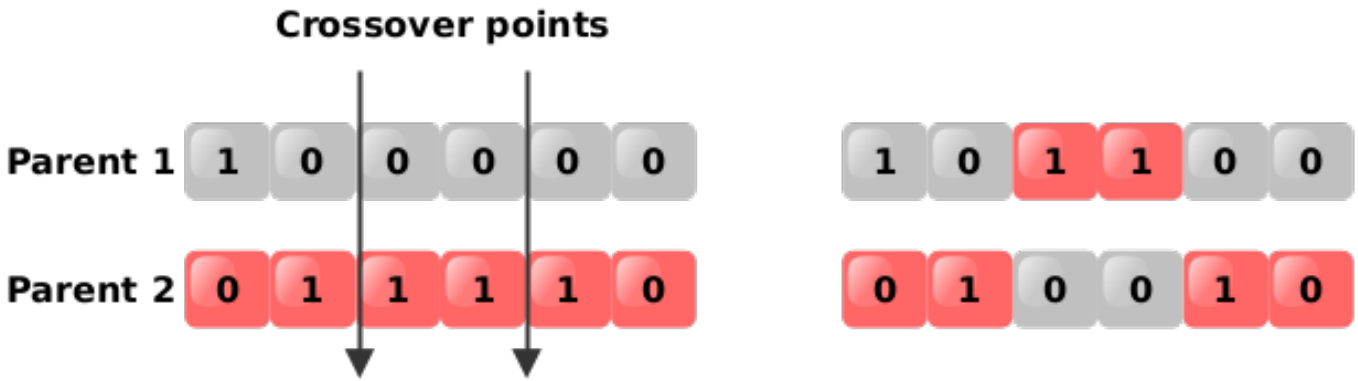


Figure: Multipoint crossover operation with $K = 2$.

Components of a Genetic Algorithm

Genetic operators

- The **uniform** crossover uses a fixed proportion to determine the contribution of each parent, and that contribution occurs at the level of the gene, not at the level of the segment. During the crossover operation, a random mask of 0 and 1 is generated according to the crossover rate. For a crossover rate of 0.5, half of the genes in the children would be inherited from parent 1, while the other half would be inherited from parent 2. The genes that correspond to bit 1 are taken from parent 1, while those corresponding to 0 are taken from parent 2 (Chaudhry and Usman, 2017).

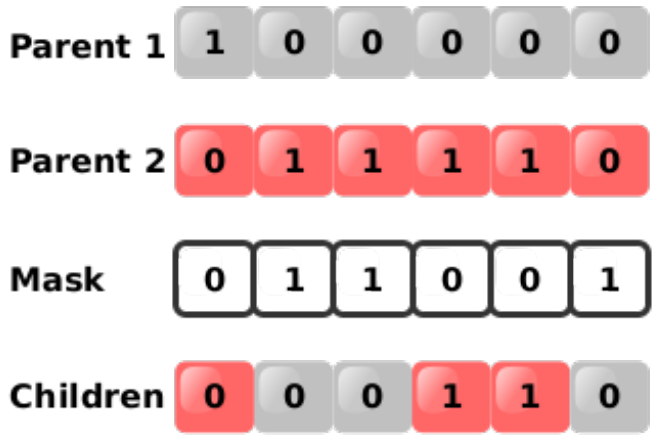


Figure: Uniform crossover operation (Adapted from Chaudhry and Usman, 2017).

Components of a Genetic Algorithm

Genetic operators

- The mutation operator occurs by randomly altering some genetic characteristics of certain individuals that have been selected by a probabilistic criterion (Goldberg et al., 1989).
- The mutation is an operation that uses only one individual (parent) to create another individual (child), applying some kind of random modification to its representation (Eiben and Smith, 2007).
- Several types of mutations are described by Soni and Kumar (2014): **insertion**, **inversion** and **uniform mutation**.

Components of a Genetic Algorithm

Genetic operators

- The **insertion** mutation selects two random genes from the individual and then moves the first gene to follow the second, moving all the other genes accordingly with this change. This type of mutation does not change much the order in which the genes appear and is used in permutation problems.

Figure: Insertion mutation operation.

Components of a Genetic Algorithm

Genetic operators

- In the **inversion** mutation, two random genes are chosen, and to perform the operation, the inversion of all genes among the chosen ones is performed. This causes the adjacent information between the genes to be preserved, however, the order information is lost. It is also used in permutation problems.

Figure: Inversion mutation operation.

Components of a Genetic Algorithm

Genetic operators

- The **uniform** mutation changes a random gene according to a specific value that this gene can assume. That is, a gene *G* chosen to undergo the mutation can receive value *i*, where *i* corresponds to an element of the set of values that *G* can assume. This type of mutation is used in cases of representation of the real and integer type of individuals.

Figure: Uniform mutation operation.

Components of a Genetic Algorithm

Parameters

GAs have some parameters that directly impact their functioning. Although there are standard values in the literature to be used, the configuration of parameters is particular in the context in which the GA is inserted (Izidoro et al., 2014). Some of the parameters used in GAs are:

- **Number of generations:** it is one of the criteria for stopping an GA. When running a very small number of generations an GA may not find a satisfactory answer. On the other hand, a very large number of generations can negatively impact computational time.
- **Population size:** it is the number of individuals present in each generation of GA. It can be static, remaining the same throughout the execution of the algorithm, or it can undergo changes in its size according to the execution. Larger populations tend to consume more execution time while smaller populations may end up not covering the entire scope of the problem.

Components of a Genetic Algorithm

Parameters

- **Probability of Crossover:** it is a percentage that indicates the chance of an individual exchanging genetic material with another within the population in order to generate descendant individuals. Its purpose is to refine the search for better individuals. GAs with high crossover rates tend to introduce new characteristics to the population more quickly, but they can end up losing good individuals that can be replaced. The use of crossover operators with high or low probability depends on the context of the problem.

- **Probability of Mutation:** determines the chance of an individual changing their characteristics, and aims to prevent the GA from being trapped in local minimums, being responsible for inserting diversity into the population. It usually has low rates, but high or low rates also depend on the context.

- **Tournament Size:** it is a parameter of the selection per tournament that controls the selective pressure, avoiding an early convergence of the GA. The tournament operator works by randomly selecting N individuals from the population and selecting the best solution among them to move on to the next generation.

Components of a Genetic Algorithm

Stopping criteria

- There are two main ways to end the execution of an GA (Eiben and Smith, 2007). The first is related to the characteristics of the individuals who make up the solution to the problem. When it is possible to identify an optimal pattern about individuals in the population, there is no longer a need to continue executing the GA, thus being able to end its execution.
- The second form occurs when it is not known to identify an optimal pattern of individuals. Some factors used to stop GA execution:
  - Maximum execution time of the algorithm or number of generations is exceeded;
  - Total number of assessments made by the fitness function is achieved;
  - Improvements in individuals made through genetic operators and selection have already reached a certain limit, with no further changes.
- Generally, an GA has its execution finished when one of the ways described previously is satisfied: a certain optimum (or satisfactory) value is reached by the individuals or a stop condition is satisfied.

Example

The basic function of each operator in a GA:

- **Selection:** Algorithm guide for promising areas of the search space.
- **Crossover:** Changes in the context of information.
- **Mutation:** Introduces innovation.
- **Elitism:** The best individual of each generation is copied unchanged for a next generation.

Example

- Maximize the function $f(x) = x^2$.
- Individual size = 5.
- Population size = 4.
- Generate initial population: Assign randomly 1s and 0s.

Example

x	População	$f(x)=x^2$	fi(x) S fi(x)	Roleta
26	11010	676	0.42	3
4	00100	16	0.01	0
27	11011	729	0.45	0
14	01110	196	0.12	1
Somatório ->		1617		

Pool	Mate	Cruzamento	Mutação	Nova População	x
11010	2	2	-----	11010	26
11010	3	1	---3-	11100	28
11010	0	2	---3-	11000	24
01110	1	1	-----	01010	10

Legenda -> Cruzamento Mutação Sem Cruzamento Continua? <S/N>

Evaluating Results

Some Tips

- Do not be hasty when evaluating the results. You need to test several times (For simple problems at least 30).
- Use statistical measures (averages, medians, standard deviation, etc.).
- Save as much information as possible about your population: Average, Best, and Worst Fitness every generation.
- Draw charts to track the progress of these variables.
- Compare with a random search algorithm.

Evaluating Results

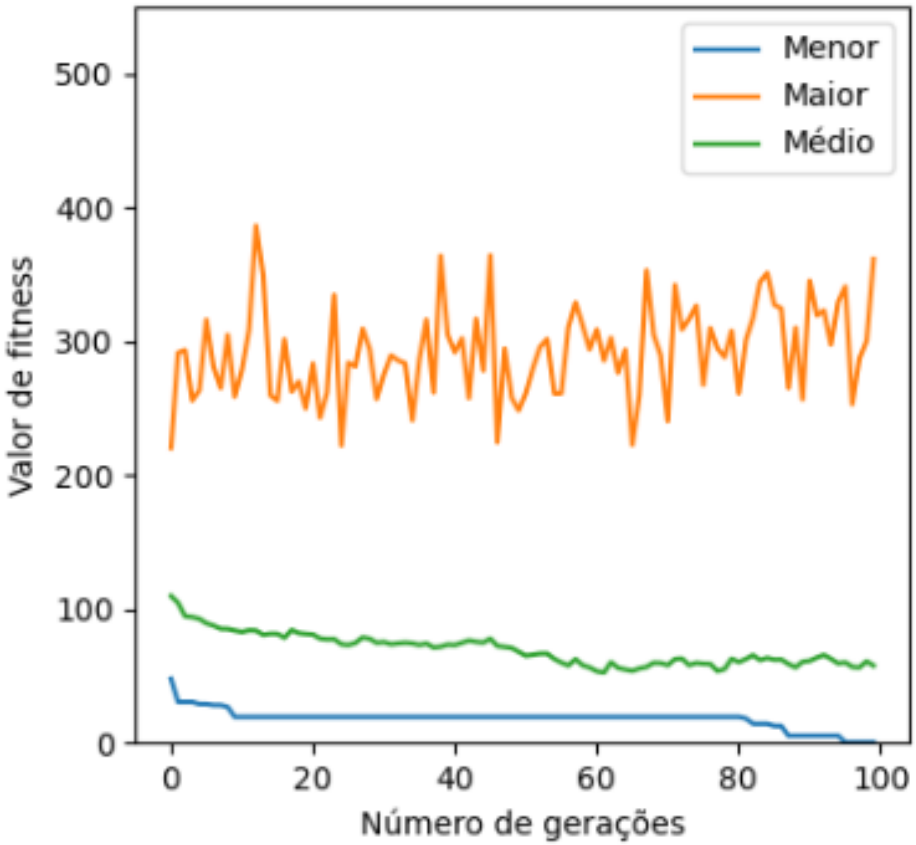


Figure: Average, Best, and Worst Fitness every generation.

Parallel Genetic Algorithms

Parallel Genetic Algorithms

- There are cases where running GAs on serial machines can take days or even weeks to complete their execution, and a parallel approach can bring considerable gains in execution time and resource utilization (Umbarkar and Joshi, 2013).
- Parallel computing refers to several processes that work simultaneously to solve a given problem. Parallelism works by decomposing the workload, or tasks, between the various computational resources available, to make gains in terms of time and/or improvement in results.
- Problem approaches that use parallelism must take communication between processes into account, as often just adjusting a serial problem does not guarantee the best parallel approach (Madhuri and Deep, 2009).

Parallel Genetic Algorithms

- GAs have an implicit characteristic of a naturally parallel search for a solution. This is evidenced by noting that each individual within a population seeks to optimize their fitness alone (Cantu-Paz, 1998).
- It is this property that allows in the same population, with all individuals exposed to the same operators, the emergence of different good solutions. This fact links the concept of GAs to the concept of parallelization, intuitively indicating the idea of a parallel GA.
- With this in mind, parallel GAs work, for example, with multi-population problems, where several different processes work independently with their respective populations and GAs. At the end of each parallel execution or even after a few generations, processes can exchange messages with each other, sharing and integrating solutions (Majd et al., 2013).

Parallel Genetic Algorithms

- Although sequential GAs have shown success in several different contexts and problems, there are cases where this approach is not enough, and it is necessary to move to parallel implementation of GA. It can be cited as cases where parallel GAs are more advantageous (Nowostawski and Poli, 1999):
  - When the population is too large;
  - When the fitness function consumes a lot of resources (time and/or memory);
  - When sequential GAs fall into suboptimal regions of the search space.

Parallel Genetic Algorithms

- There are three general classes of parallel GAs: master-slave, coarse-grained (island) and fine-grained (cell) (Fauzi Mohd Johar et al., 2013).
- Parallel GAs of the master-slave type use the same idea as a sequential GA and apply parallelism in a simple way, where neither the genetic operators are altered nor restricted.
- In the master-slave model, generally, only the evaluation of individuals by the fitness function is done in parallel, all the control of generations, selection and mutation and crossing operators are still done serially.
- This type of implementation is sought mainly when fitness analysis is complex and consumes a lot of computational resources.
- A disadvantage of this method is that the master process is often idle, waiting for the other processes (Fauzi Mohd Johar et al., 2013).

Parallel Genetic Algorithms

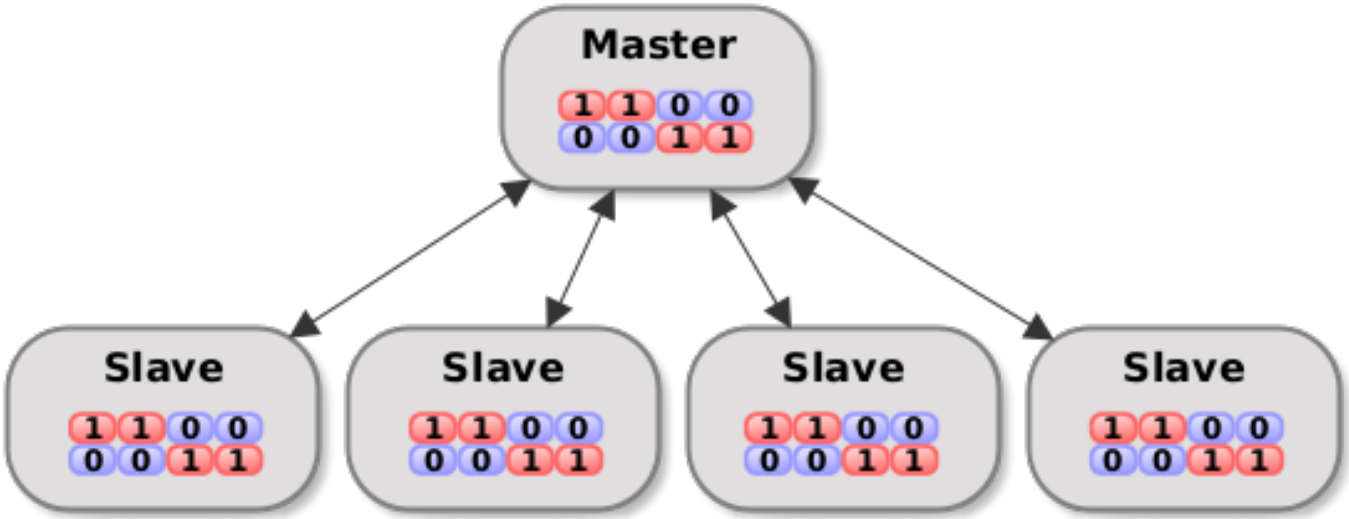


Figure: Example of a master-slave GA (Fauzi Mohd Johar et al., 2013).

Parallel Genetic Algorithms

- Another type of parallel approach in GAs is coarse-grained. This form presents a significant difference in the modelling of an GA, since it is the parallelization of populations, dividing into subgroups that are directed to different processes (islands).
- Each island has its own evolutionary process, with its respective generations and operators of selection, crossing and mutation. The processes work as a sequential GA, separately, and during the execution of the processes, individuals from one island migrate to another in order to generate greater genetic variability (Madhuri and Deep, 2009).
- Finally, there are the parallel GAs of the fine-grained or cellular type, where individuals are distributed in different processes and the crossing operations are restricted to neighbouring processes (or cells).
- Parallel-fine-grained GAs require a well-defined topology and generally decline in performance as the population increases (Fauzi Mohd Johar et al., 2013). This is a viable alternative mainly when implemented on a SIMD (Single Instruction Multiple Data) processing device.

References

References

- Cantu-Paz, E. (1998). A survey of parallel genetic algorithms. CALCULATEURS PARALLELES, 10.
- Camargo, G. d. M. (2006). Controle da pressão seletiva em algoritmo genético aplicado a otimização de demanda em infra-estrutura aeronáutica. Master's thesis, Escola Politécnica, Universidade de São Paulo.
- Chaudhry, I. A., e Usman, M. (2017). Integrated process planning and scheduling using genetic algorithms, Tehnički vjesnik, 24(5), pp. 1401-1409. https://doi.org/10.17559/TV-20151121212910
- Dréo, J.; Chatterjee, A.; Pétrowski, A.; Siarry, P. e Taillard, E. (2006). Metaheuristics for Hard Optimization: Methods and Case Studies. Springer Berlin Heidelberg.
- Dréo, J.; Chatterjee, A.; Pétrowski, A.; Siarry, P. e Taillard, E. (2006). Metaheuristics for Hard Optimization: Methods and Case Studies. Springer Berlin Heidelberg.
- Eiben, A. E. e Smith, J. E. (2007). Introduction to Evolutionary Computing . Springer Verlag.

References

- Fauzi Mohd Johar; Farah Ayuni Azmin; Mohamad Kadim Suaidi; Shibghatullah, A. S.; Badrul Hisham Ahmad; Siti Nadzirah Salleh; Mohamad Zoinol Abidin Abd Aziz e Shukor, M. M. (2013). A review of genetic algorithms and parallel genetic algorithms on graphics processing unit (gpu). In 2013 IEEE International Conference on Control System, Computing and Engineering, pp. 264–269.

- Goldberg, D.; David Edward, G.; Goldberg, D. e Goldberg, V. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Artificial Intelligence. Addison-Wesley Publishing Company.

- Izidoro, S. C.; de Melo-Minardi, R. C. e Pappa, G. L. (2014). GASS: identifying enzyme active sites with genetic algorithms. Bioinformatics, 31(6):864–870.

- Katoch, S.; Chauhan, S. S. e Kumar, V. (2020). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications.

- Madhuri e Deep, K. (2009). A state-of-the-art review of population-based parallel meta-heuristics. In 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), pp. 1604–1607.

References

- Majd, A.; Lotfi, S. e Sahebi, G. (2013). Review on parallel evolutionary computing and introduce three general framework to parallelize all ec algorithms. The 5th Conference on Information and Knowledge Technology, pp. 61–66.
- Meadows, B.; Riddle, P.; Skinner, C. e Barley, M. M. (2013). Evaluating the seeding genetic algorithm. In Cranefield, S. e Nayak, A., editores, AI 2013: Advances in Artificial Intelligence, pp. 221–227, Cham. Springer International Publishing.
- Mitchell, M. (1998). An Introduction to Genetic Algorithms. A Bradford book. Bradford Books.
- Nowostawski, M. e Poli, R. (1999). Parallel genetic algorithm taxonomy. In 1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No.99TH8410), pp. 88–92.
- Soni, N. e Kumar, T. (2014). Study of various mutation operators in genetic algorithms. volume 5, pp. 4519–4521.
- Umbarkar, A. J. e Joshi, M. S. (2013). Review of parallel genetic algorithm based on computing paradigm and diversity in search space. ICTACT Journal on Soft Computing, 3(4):615–622.
- Umbarkar, A. J. , P. D. S. (2015). Crossover operators in genetic algorithms:a review. ICTACT Journal on Soft Computing, 6(1):1083–1092.