1 Introduction to Genetic Algorithms

2 Theory

2.1 Basic Components

The following are the most basic components that are present in most genetic algorithms:

1. Fitness function for optimization.
2. Population of chromosomes.
3. Selection of chromosomes for reproduction.
4. Crossover to produce the following generation of chromosomes.
5. Random mutation of chromosomes in the following generation.

2.1.1 Fitness Function

Fitness is a term from biology which defines the extent to which a certain type of organism is able to pass itself onto the next generation, influenced by how well a given organism does the job it was evolved to do. Say one beaver has bigger teeth than the other, making it faster in cutting down trees. Based on this, the beaver with the bigger teeth has the better 'fit', making it more likely it will pass down its genes.

In the context of genetic algorithms, the fitness function is responsible for making the algorithm able to achieve the purpose it was made for. The fitness function determines how well the evolving chromosomes fit to the goal of the objective over a range of scores. The direction in which the population evolves is based on these scores.

The fitness function is also known an an **objective function**.

2.1.2 Chromosomes

Usually, the goal of a genetic algorithm is to solve a problem by making units learn what the goal value is. Chromosomes can thus be described as the values an individual unit contains, that are used to define the learning values for individual units. It is a way to represent the learning values.

For each individual unit, their corresponding chromosomes are defined in an arrary that contains parameters. The representation of the values that the parameters define is dependent on the problem that the creator is trying to solve. A chromosome can be defined as following:

$$\text{chromosome} = [p_1 ... p_N]$$

Where N is the number of dimensions of the problem and p corresponds to the given parameters. p can be expressed in, for example, binary, real numbers and so forth.

2.1.3 Selection Operator

When producing the next generation of the population, their attributes will need to change to see improvement after reproduction. As chromosomes were discussed, they contain the properties individuals in a population hold. These chromosomes need to be slightly changed, while taking the fitness score into consideration, to see improvement in the next generation of the population. This is also called mutation, where the values of chromosomes are changed, be it slightly or radically. The methods at which selection operators work will be discussed in the following section.

2.2 Convergence

2.3 Local optima

3 Weighing Methods

For this section, the sequence of and underlying methods for building a genetic algorithm will be discussed.

3.1 Initialization

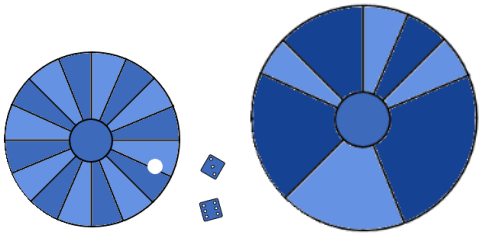
To start off, the represenation of the first generation needs to be determined. Often, the properties of the first population will be randomly generated. This makes it possible for the algorithm to find solutions over a wide range within the search space (search space defines the set of all possible points within the search space). However, when presumptions can be made beforehand about the possible solutions, the properties can be seeded with areas in which an optimal solution is likely to be found.

3.2 Parent Selection

For this section, a couple of methods will be discussed for selecting parents from the population that can reproduce. Each method will have their respective pros and cons.

3.2.1 Roulette Wheel

The parents are chosen with a probability proportional to their fitness. Usually when someone spins a roulette wheel, the outcome will be entirely random, since each slot in the wheel will have the same probability of being chosen. However, when implementing a roulette wheel for parent selection in a genetic algorithm, the probability can be adjusted based on the fitness of the parents. With this, it is ensured that only the better performing parents are chosen for reproduction. This, however, does not guarantee that worse performing parents can be selected, since any parent still has a chance of being chosen. The probability increase in relation to the fitness value can ofcourse be defined in the program.



(a) Roulette wheel (b) Weighted

Figure 1: Comparison between a regular roulette wheel and a weighted roulette wheel

3.2.2 Elitism

With elitism, the agents with the best fitness scores are guaranteed to be carried over. This comes with the benefit that these agents are not lost between generations, so that high quality properties are preserved. This can result in finding a solution much earlier due to the preservation of good solution answers. It can also make the population more stable, since good-fit answers will not be lost between generations.Elitism can also introduce some negatives. Mainly in reducing population diversity by preserving the best genetics. This can result in a smaller search space being explored, meaning other solutions are less likely to be considered.

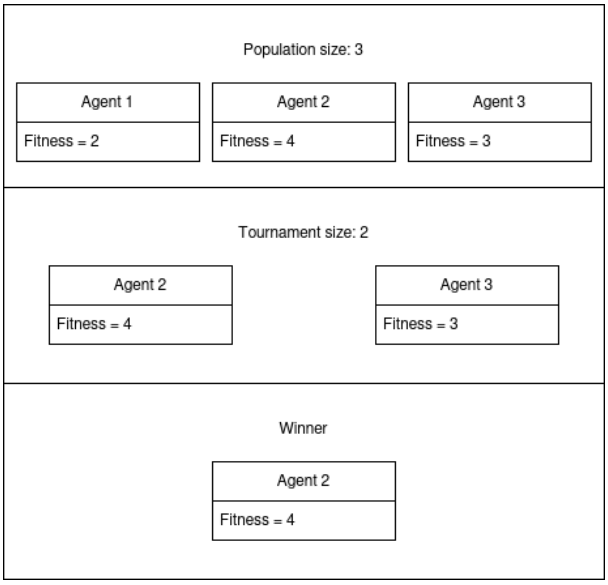


Figure 2: Overview of a tournament selection

3.2.3 Tournament Selection

A subset is randomly chosen, and the best is selected. When using tournament selection before crossover, the population will be divided into subsets. Inside each tournament (subset) the best performing agent will be chosen for reproduction. For this method, it is relatively simple to control the behaviour of, since the tournament size can be tweaked by the user. The way it gives less fit individuals a change to reproduce is also favourable for introducing variety in the population.However, if the tournament size parameter is set too big, it can lead to premature convergence, where diversity in the population is lost. Also the fact that agents that are more fit are more likely to succeed, diversity can be lost, which can prevent exploration of the search space.

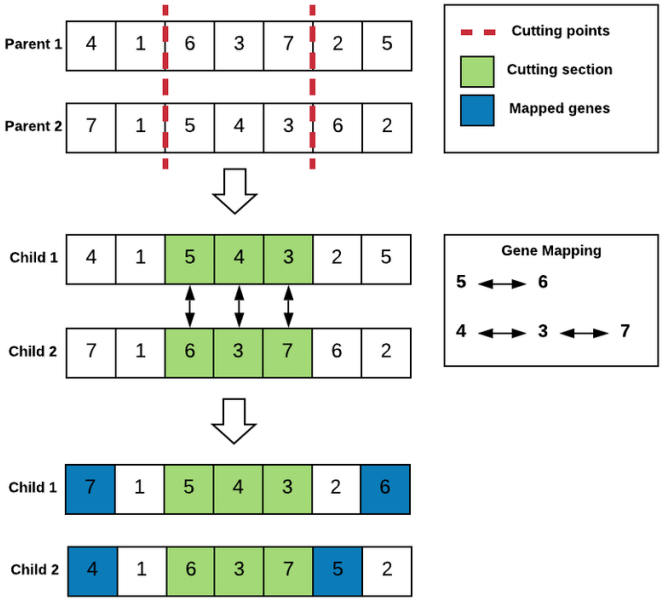


Figure 3: Overview of partially mapped crossover.

3.3 Crossover

Genetic operator to combine information of parents or use cloning. Child can be subject to mutation, see next subsection. Typically, the genetic structure consists of data structures, like arrays of bits, vectors and arrays.

In the following subsections, two methods of crossover relevant to the job sequencing problem, order and partially mapped crossover, will be discussed.

3.3.1 Order Crossover

The main goal of order crossover is to combine the genetic properties of two parents while maintaining the order of their genes. Like any crossover function, two parents are selected for combining their genetic properties to make a new child. A segment of the first parents' genetic properties are selected and will be copied to the child into the same position in its' genetic properties. Then, the remaining elements of the second parent are crossed over to the child, while keeping the order of the properties of the second parents' genetics.

The benefits of using order crossover are the preservation of order of the parents' genetic sequence, it also introduces diversity in the offspring. This method can however be more difficult to implement than other crossover methods, like single point crossover, which simply swaps part of the genetic sequences of parents.

3.3.2 Partially Mapped Crossover

With partially mapped crossover, the combination of two parents' permutations takes place with preserving order and uniquess in mind. The way it works is similar to order crossover, where a segment of the first parents' genetic sequence is taken and placed in order for the genetic sequence of the child. See figure 3.

However with partially mapped crossover, the values between parents are mapped, so that it is ensured that the offspring contains valid permutations. So, when creating a child, the segment of the first parent is copied over the the child. Next, the remaining positions in the offspring are filled with elements from the second parent, but if an element from the second parent is already present in the offspring (due to the mapping), it is replaced by the corresponding mapped element from the first parent.

3.4 Mutation

3.4.1 Swap Mutation

3.4.2 Random Resetting

Problems: - Genetic drift

4 Conclusion

5 Bibliography

Carr, J. (2014). An Introduction to Genetic Algorithms.

https://www.whitman.edu/Documents/Academics/Mathematics/2014/carrjk.pdf

Jannoud, I., Jaradat, Y., Masoud, M. Z., Manasrah, A., Alia, M. (2021). The Role of Genetic Algorithm Selection Operators in Extending WSN Stability Period: A Comparative Study. Electronics, 11(1), 28. https://doi.org/10.3390/electronics11010028