1 Theory of Genetic Algorithms

Genetic algorithms are optimization techniques inspired by the process of natural selection. They work by evolving a population of potential solutions through iterative improvements based on methods such as selection, crossover, and mutation. By mimicking biological evolution, genetic algorithms can efficiently explore complex search spaces to find optimal or near-optimal solutions. This section outlines the fundamental components of genetic algorithms.

1.1 Basic Components

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

1. Fitness function for optimization.
2. Population of agents, each agent containing a sequence of genetics/chromosomes.
3. Parent selection for reproduction.
4. Crossover to produce the following generation of chromosomes.
5. Random mutation of chromosomes in the following generation.

These components work together to simulate the evolutionary process. In the population, two parents will "cross over" their genetics to create a new child, where the aim is to iteratively improve candidate solutions toward a solution that is near-optimal to the given problem. Individuals in the population will be referred to as "agents".

1.1.1 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 fact, the beaver with the bigger teeth is better 'fit' to deal with its' environment, 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 goal 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, among other methods like crossover and parent selection.

1.1.2 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. This is sometimes referred to as a genetic sequence.

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 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.

1.1.3 Parent selection

Before reproducing, agents will need to be selected for reproduction. A seemingly simple step, it is usually crucial for maintaining agents with a good fit, while still introducing some variety in the population. When parents are selected entirely at random, agents who have genetics that are close to the solution can be lost in this process. However, when only choosing agents with a good fitness, it can occur that the entire population will move to a solution that might not be the best one. Doing this will decrease diversity in the population, also called reaching "convergence" prematurely. This is why it is often chosen to promote agents with good fitness, while still giving agents with a lesser fitness a chance to reproduce.

1.1.4 Crossover

Crossover occurs when two parents reproduce to make a child, where the genetic sequence of the parents are combined in some form. This can involve swapping sections of the sequence, moving genetics around and more. In the next chapter, order and partially mapped crossover will be discussed.

1.1.5 Mutation

When producing the next generation of the population, their attributes can be changed to introduce more variety/randomness in the genetics of the population, also called "mutation". With mutation, a random agent or a group of agents will be chosen for mutation, where information in their genetic sequence(s) will be changed by, for instance, swapping a gene with another or replacing them with a random gene. Randomness is important for genetic algorithms, since they, like stated before, can prevent premature convergence without exploring enough of the solution space.

1.2 Convergence

Convergence refers to the extent to which a genetic algorithm approaches a solution or a set of solutions over a span of time. When the algorithm is correctly implemented, convergence means that the population is moving towards finding a solution, which can be noted by the individuals within the population becoming more similar to eachother. When the average fitness is climbing is usually indicative of the population moving toward convergence.

In the ideal situation, the population will move to convergence for a "global optimum", meaning that it is moving towards the best possible solution in the entire solution space. However, more often an algorithm will move to convergence for a "local optimum", meaning that it is moving towards the best possible solution compared to neighbouring solutions. This can be seen as finding a "good" answer, but not the "best" answer. More often than not, finding the global optimum would be too time intensive to be computed. This is why finding a good-enough answer is the general goal for most genetic algorithms.

1.2.1 Premature Convergence

When a genetic algorithm finds an answer without exploring enough of the solution space, it will often lead to a suboptimal answer. This is also called "premature convergence". This is often due to the properties of the population becoming too similar too quick, which leads to getting stuck in a local optimum without being able to "see" other optima. When diversity is low, the genetic algorithm may fail to explore other promising areas of the solution space.

Since the algorithm provided a solution before experiencing enough variation, finding an optimal method for introducing enough variety into the population is often what makes or breaks a genetic algorithm. In the section below, different methods for introducing variation into the population will be discussed.

2 Weighing Methods

For this section, the sequence of and underlying methods for building a genetic algorithm will be discussed. This includes different methods used for parent selection, crossover and mutation.

2.1 Initialization

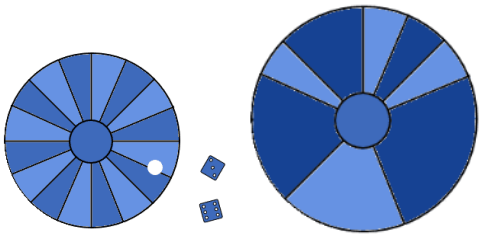
To start off, the represenation of the first generation needs to be determined. Often, the properties of the first population will be randomly generated. This makes it possible for the algorithm to find solutions over a wide range within the solution space. However, when presumptions can be made beforehand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

2.2 Parent Selection

For this section, a couple of methods will be discussed for selecting parents from the population that can reproduce.

2.2.1 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, parents with a better fitness are more likely to be selected for crossover. This, however, does not guarantee that parents with a lesser fitness can be selected for reproduction, since any parent still has a chance of being chosen. This is not a bad idea though, since this ensures that it is more likely that the population will be more diverse, possibly preventing premature convergence.



(a) Roulette wheel (b) Weighted

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

2.2.2 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 solution space being explored, meaning other solutions are less likely to be considered.

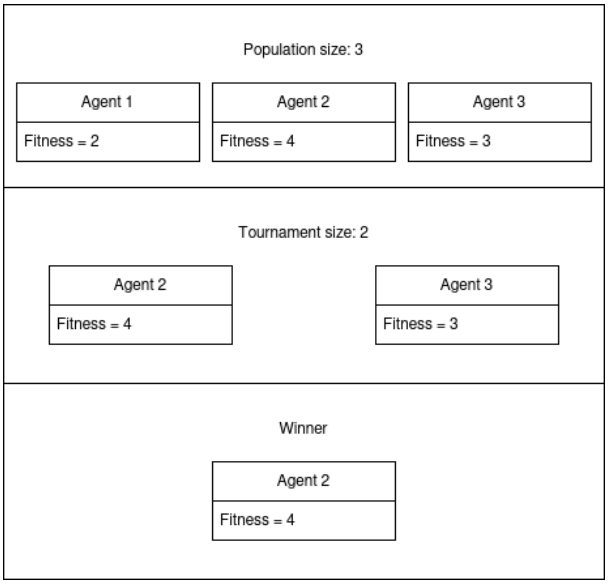


Figure 2: Overview of a tournament selection

2.2.3 Tournament Selection

For tournament selection, a subset of the population is chosen, and the agent with the best fitness is selected for reproduction. 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 by creating subsets 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 solution space.

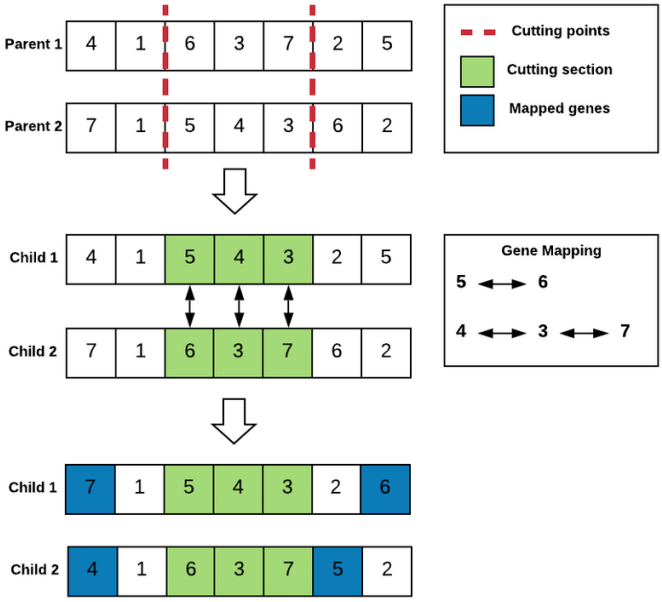


Figure 3: Overview of partially mapped crossover.

2.3 Crossover

Crossover is the genetic operator for combining information of parents. In the following subsections, two methods of crossover relevant to the job sequencing problem, order and partially mapped crossover, will be discussed.

2.3.1 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, while 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.

2.3.2 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 **??**. 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.

2.4 Mutation

2.4.1 Swap Mutation

Swap mutation is one of the most basic mutation functions for genetic algorithms. It goes as follows:

1. An agent is selected for mutation.
2. Two positions in the gene sequence are selected.
3. The two selected genes are then swapped over.

The benefit of using swap mutation is that it introduces new sequences to the population, while maintaining valid permutations, meaning that all elements of the reference set/schedule only appear once in the genetic sequence.

2.4.2 Random Resetting

With random resetting, one or more positions within the chosen individual are randomly selected for resetting. These positions can either be chosen at random or predetetermined. The selected positions then get replaced by either a new random value, or from a subset of available values chosen for resetting. Just like swap mutation, random resetting can introduce diversity to the population. Both methods are suitable for preventing premature convergence to local optima, since more diverse solutions are explored this way. By introducing new values, random resetting is suitable for expanding the exploration of new solutions in the solution space.

3 Conclusion

Genetic algorithms are a powerful optimization technique inspired by natural selection. By leveraging functions such as fitness functions, selection operators, crossover, and mutation, these algorithms efficiently explore complex solution spaces. The balance between exploitation (choosing the best solutions) and exploration (introducing diversity) is crucial for achieving optimal or near-optimal results.

One of the main challenges in genetic algorithms is premature convergence, where a population gets stuck in a local optimum without exploring alternative solutions. Strategies such as diverse parent selection and population help prevent this from occurring.

While genetic algorithms may not always guarantee the absolute best solution, their ability to provide high-quality estimates in a reasonable time makes them a valuable tool in various fields, from machine learning to scheduling problems and robotics.

4 Bibliography