International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 06 | June - 2021 ISSN: 2582-3930

Introduction to Genetic Algorithms: Models and Applications

Satish Kumar Gupta
Assistant Professor, Department of Information Technology,
Gopal Narayan Singh University, Jamuhar Sasaram (Rohtas), Bihar-821305

Abstract:
The Genetic Algorithm (GA) is a genetic and natural selection and search-based optimization technique. It works for finding the optimal or near-optimal solutions to complex problems to solve. It's widely used in science and with Artificial Intelligence to solve optimization problems. This Paper also contains the detailed information about the fitness function, selection, crossover and Mutation. The aim of this Paper is to look at the Genetic Algorithm and the need for it in various functions. This paper also focuses on explaining the various models that have been adopted while using the Genetic Algorithm (GA).

Keywords: Function, Presentation, Selection, Population

Genetic Algorithm:

"Genetic Algorithms are good at exploring big, potentially enormous search spaces, searching for ideal combinations of items that you may not find in a lifetime."

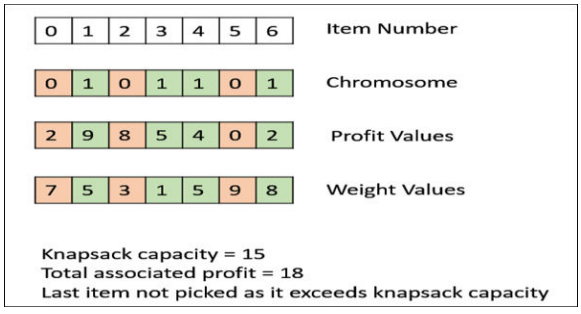
Fitness Function in Genetic Algorithm:

Simply put, the fitness function takes a candidate solution to a problem as input and outputs how "fit" or "right" the solution is for the problem at hand. In a GA, since the fitness value calculation is done many times, it must be fast. A GA can be slowed down by a slow measurement of the fitness value. Since the goal is to optimize or decrease the given objective function, the fitness function and the objective function are usually the same. An Algorithm Designer may choose a different fitness function for more complex problems with multiple goals and constraints.

The following are attributes that a fitness feature should have,

1. It must quantify how to Individuals can be created to suit a given solution or to fit a given solution it and it must be easy to compute.
2. Owing to the inherent complexity of the problem at hand, measuring the fitness function directly in certain situations might not be feasible.

In these situations, we approximate fitness to meet our requirements. The fitness equation for a 0/1 Knapsack solution is shown in the image below. It is an essential fitness feature that adds the benefit values of the selected products (which all start with a 1) and scans from the left to the right before the knapsack is full.



Parent Selection:

The process of selecting parents to mate and recombine to produce offspring for the next generation is called parent selection. Parents' choices are crucial to the GA's convergence rate since good parents encourage their children to search out new and more appropriate solutions. However, strict caution should be exercised to prevent a single from sweeping the whole population in a few years to an incredibly fit solution, as this would result in the solutions in the solution room becoming too close to one another, reducing diversity. Maintaining a high level of population diversity is critical to a GA's performance. Premature convergence is when one highly fit solution consumes the entire population; In a GA, this is a bad situation.

Appropriate Fitness Selection:

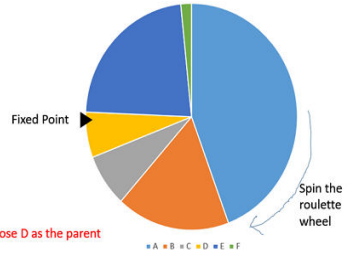
Health and fitness one of the most popular parent selection strategies is proportionate selection. Every individual has a proportional chance of becoming a parent. As a result, fitter people are more likely to marry and pass on their characteristics to the next generation. As a result, a selection policy like this puts pressure on candidates the population's fittest individuals, resulting in the evolution of better individuals over time.

Health and fitness one of the most popular parent selection strategies is proportionate selection. Every individual has a roughly equal chance of becoming a parent. Fitter people are more likely to marry and pass on their traits to the next generation as a result. As a consequence, a program like this places candidates under much stress.

Fitness proportionate selection can be implemented in two ways.

Selection of the Roulette Wheel:

The circular wheel is broken in a roulette wheel array, as previously stated. The wheel is rotated after selecting a fixed point on the diameter, as shown. The parent is the area of the wheel in front of the fixed point. The same procedures are followed for the second parent.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

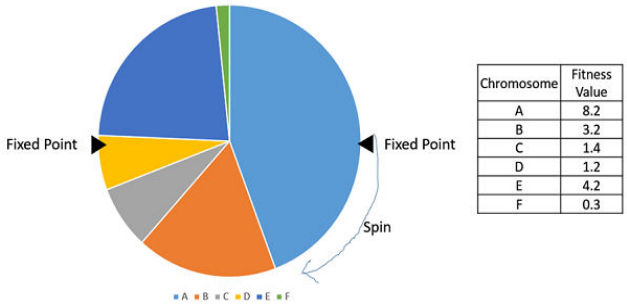
When the wheel is turned, a fitter individual gets a bigger pie on the wheel and, as a result, a higher chance of landing on the fixed spot. As a result, the likelihood of selecting a person is directly proportional to its fitness.

In terms of implementation, we use the steps below.

- Substitute S for the amount of the finesses.
- Choose a number between 0 and S at random.
- Apply the finesses to the part number P until it hits PS, counting down from the top of the population P must be greater than S to be selected.

Stochastic Universal Sampling (SUS):

Stochastic Universal Sampling is similar to Roulette wheel array, except we have multiple fixed points instead of just one, as seen in the image below. As a consequence, in a single wheel spin, both parents are chosen. Furthermore, such a system allows for at least one selection of exceptionally fit individuals.

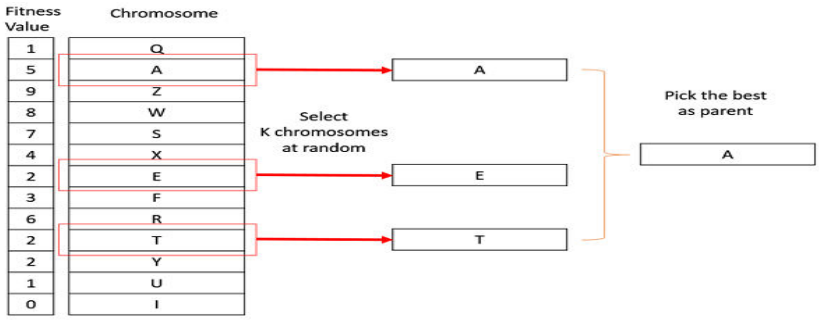


Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

It is worth noting that fitness proportionate selection strategies do not work in situations where fitness can be damaging.

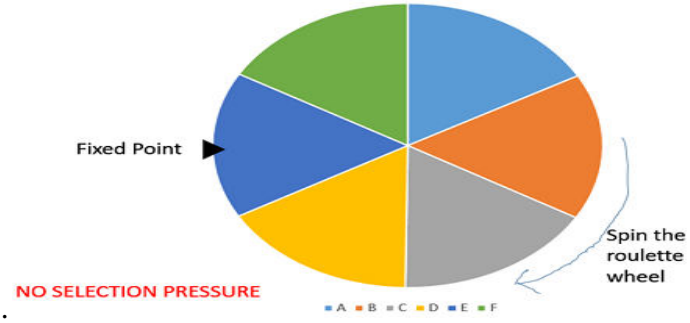
Tournament Selection:

In a K-Way tournament, we choose K people at random from the community and choose the best of them to become parents. The procedure for choosing the next parent is the same as before. Tournament Selection is also widely used in the since it can be used for negative exercise ideals, literature is useful.



Rank Selection:

Rank Selection can be used on both positive and negative fitness values, and is most often used where the population's fitness values are somewhat similar (this usually happens at the end of the run). As seen in the diagram, each person has a nearly equal share of the pie (as in the case of proportionate fitness selection), and hence each person, no matter how fit they are in comparison to one another on the other hand, has a nearly identical probability of being selected as a parent. As a result, selection demand for fitter individuals reduces, causing the GA to make bad parental decisions under these circumstances



Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

.

When choosing a parent, we eliminate the notion of a fitness value. Every individual in the population, on the other hand, is ranked according to their level of fitness. The parents are chosen based on each individual's rank rather than their fitness. Individuals with higher ranks are favoured over those with lower ranks.

Chromosome	Fitness Value	Rank
A	8.2	1
B	8.0	4
C	8.04	2
D	7.94	6
E	8.01	3
F	7.98	5

Selection at Random:

In this technique, we select parents at random from the current population. This strategy is usually avoided because there is no selection pressure for fitter people.

Basics of Crossover:

The crossover operator is similar to reproduction and biological crossover. Many parents are selected, and one or more offspring are produced using the parents' genetic material. In a high-probability GA – pc, the crossover is commonly used.

Crossover Operators:

We'll look at some of the most well-known crossover operators in this segment. It's worth noting that these crossover operators are fairly broad, and the GA Designer may choose to use a more problem-specific crossover operator instead.

One Point Crossover:

A random crossing point is chosen in this one-point crossover, and the tails of the two parents are switched to produce new offspring

.

Multi-Point Crossover:

A one-point crossover variant in which alternating segments are switched to produce new offspring is known as multi-point crossover.

.

Uniform Crossover:

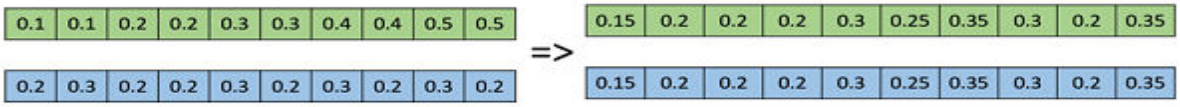
We don't partition the chromosome into segments using a uniform crossover; instead, each gene is treated separately. In this step, we basically flip a coin to see whether each chromosome will be transmitted on to the children. We may also skew the coin in one parent's favour, resulting in a child with more of that parent's genetic material

Whole Arithmetic Recombination:

This is a straightforward integer representation that uses the formulae below to calculate the weighted average of the two parents.

- Child1 = α.a + (1-α).b
- Child2 = α.a + (1-α).b

If α = 0.5, as seen in the example below, all children would be similar.



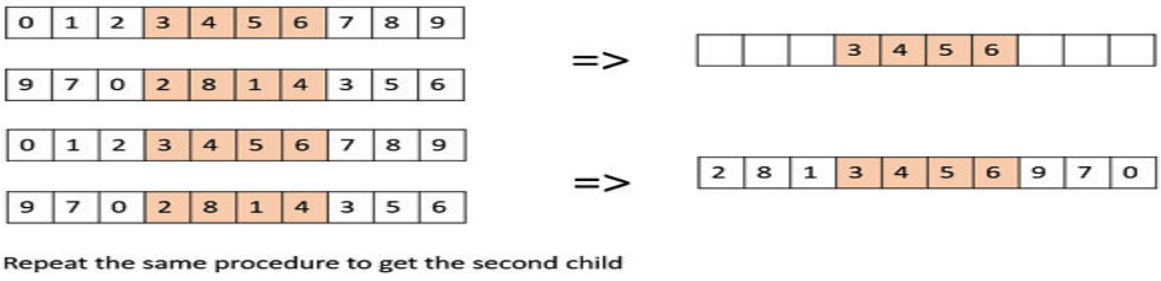
Davis' Order Crossover (OX1):

OX1 is used in permutation-based crossovers to relay relative ordering information to the offspring. Here is how it works:

• Copy the segment from the first parent to the first offspring by making two random crossover points in the parent.

• Copy the remaining unused numbers from the second parent to the first brother, wrapping around the set at the second crossover point in the second parent.

• Reverse the parent-child relationship with the second child.



Repeat the same procedure to get the second child

Shuffle Crossover, Order-based Crossover (OX2), Partially Mapped Crossover (PMX), and Ring Crossover are just a few of the other crossover available.

Introduction to Mutation:

A mutation is a minor random alteration in the chromosome that results in a new solution in simple terms. It's used to keep and grow a population's genetic diversity, and it's normally done with a low chance – pm. The GA is reduced to a random search if the likelihood is very large.

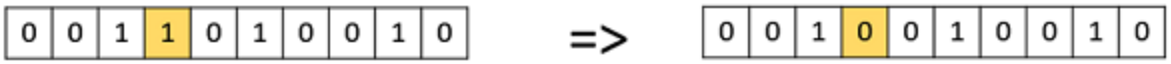
The part of the GA that deals with the "exploration" of the search room is known as the mutation. GA convergence does not require a crossover, but mutation does.

Mutation Operators:

In this section, we will go by using some of the most commonly used mutation operators This isn't an exhaustive list; the GA designer can discover that merging these methods or using a problem-specific mutation operator will solve the problem is more beneficial.

Bit-Flip Mutation:

We randomly choose one or more bits to flip in this bit-flip mutation. For binary encoded GAs, this is used.

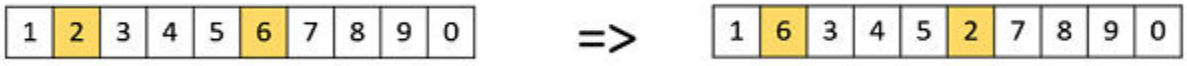


Random Resetting:

Random Resetting is a bit flip extension that uses an integer representation. A random value is assigned to a gene selected at random from the set of allowable values in this method.

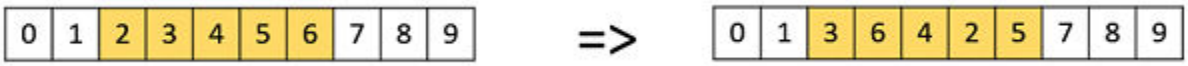
Swap Mutation:

Swap mutations entail randomly selecting two chromosome positions and exchanging their values. This is a problem with permutation-based encodings**.**



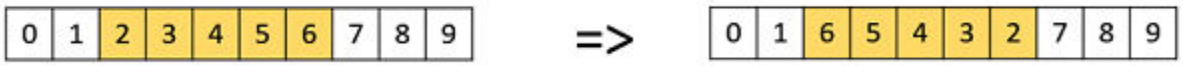
Scramble Mutation:

With permutation representations, scramble mutation is also common. A subset of genes is selected from the entire chromosome, and their values are scrambled or shuffled at random.



Inversion Mutation:

Inversion mutation is similar to scramble mutation in that it selects It inverts the whole string within the subset rather than shuffling it.



Survivor Selection:

The Survivor Selection Policy decides who is ejected from the next generation and who is retained. It is important because it ensures that the fittest people do not get pushed out of the community while also maintaining population diversity.

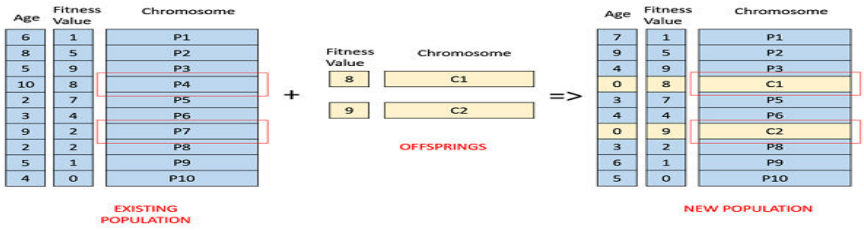
Some GAs use elitism, simply put, the population's fittest individual is often passed on to the next generation. As a result, in no conditions will the fittest member of the current population be replaced.

The most basic policy is to expel arbitrary population members; however, Since this strategy often leads to convergence problems, the following techniques are often used.

Age-Based Selection:

We do not have a definition of fitness in Age-Based Selection. It is founded on the idea that each human is allowed to stay in the population for a finite generation to reproduce before being kicked out, regardless of how fit they are.

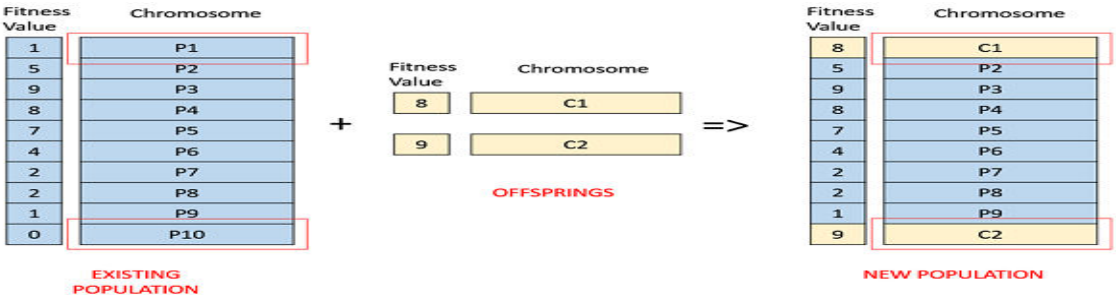
In the example below, age refers to the number of generations that an individual has lived in a society. The group's oldest members, P4 and P7, are kicked out, and the remaining members' ages are increased by one year.



Fitness Based Selection:

Children tend to replace the population's least stable people through this fitness-based selection process. Any of the previously listed screening policies, such as tournament selection, proportionate fitness selection, and so on, can be used to identify the least suitable individuals.

The twins, for example, have taken the place of the population's least fit individuals, P1 and P10, in the illustration below. Given that the fitness values of P1 and P9 are identical, eliminating one of them from the population is purely subjective



.

Termination Condition:

When it comes to deciding when a Genetic Algorithm run will finish, the termination condition is crucial. The GA seems to advance quickly; first, Changes occur every few variations, but this continues to saturate in the later stages, with only minor changes. At the end of the sprint, we want a termination state that puts our approach close to the optimum.

Typically, At least one of the above termination provisions is upheld,

- Since X iterations, the population has remained constant.
- When a certain number of generations have passed.
- When the value of the when the target function exceeds a pre-determined limit.

For example, we keep track of the generations in a genetic algorithm by keeping a counter the population has not improved. This counter was initially set to zero.When we do not create offspring that are larger than the population's individuals, we add to the counter. The counter is reset to zero if the fitness of each of the offspring changes. When the counter reaches a preset value, the algorithm comes to a halt.

The termination condition, like other GA parameters, is a highly complex challenge, and the GA designer will experiment with different solutions to see what works best which better suits his specific problem.

Models of Lifetime Adaptation:

Anything we have spoken about so far in this tutorial has been focused on Natural selection and genetic diversity by recombination and mutation are part of the Darwinian paradigm of evolution. Just the information found in a person's genotype can be passed on to future generations in nature.

Other lifetime adaptation models, such as the Lamarckian and Baldwinian models, do exist. It should be noted that determining. It is debatable which model is the safest, and analysis shows that lifespan adaptation is extremely problem-specific.

As in Genetic Algorithms, we always pair a GA with a local search. Of such cases, one might use either the Lamarckian or Baldwinian Model to determine what to do with the people found through a local quest.

Lamarckian Model:

The Lamarckian Model states that traits acquired during a person's lifetime can be passed down to their descendants. It bears the name of Jean-Baptiste Lamarck, a French biologist.

Although just the genotype's knowledge can be shared, as we all know, natural biology has ignored Lamarckism. However, Applying the Lamarckian model to real issues has been shown to provide good results. A local search operator in the Lamarckian model scans the neighbourhood for new traits, and if a stronger allele is found, it becomes the offspring.

Baldwinian Model:

The Baldwinian model, named after James Mark Baldwin, is a popular choice, is a term that falls somewhere in the centre (1896). The Baldwin hypothesis predicts that chromosomes would code for a proclivity to develop good behaviours. This implies that, according to the Lamarckian model, we do not pass on acquired characteristics to future generations, nor do we ignore acquired traits completely as the Darwinian model did.The Baldwin Model is in the center, coding a person's proclivity other than the traits themselves, to learn traits.

A local search operator in this Baldwinian Model searches the neighbourhood (acquiring new traits), and if a stronger chromosome is discovered, it simply attributes the increased fitness to the chromosome rather than modifying it. And if the trait is not transmitted on to future generations, the fitness shift shows the chromosomes' ability to "acquire" it.

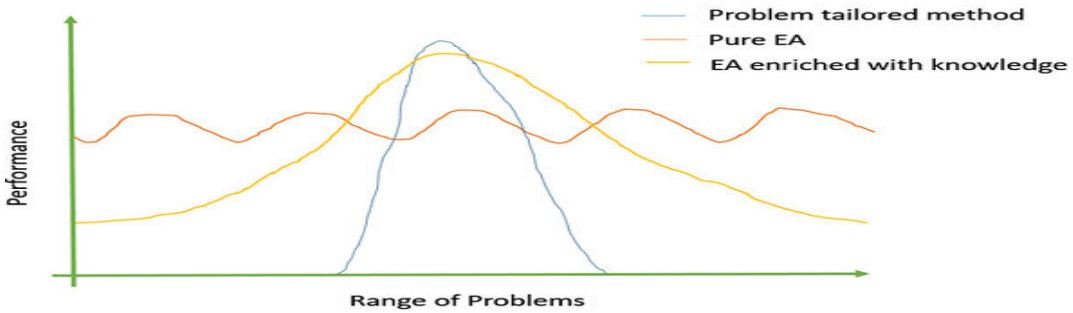
Effective Implementation:

Since GAs has such a wide variety, using them to solve any optimization problem would produce bad results.

Introduce Problem-Specific Domain Knowledge:

The more problem-specific domain information we feed into the GA, the higher the objective values we get. Custom representations, problem-specific crossover or mutation operators, and custom representations can all be used to add problem-specific information and other methods.

The picture below depicts Michalewicz's (1990) perspective on the EA.



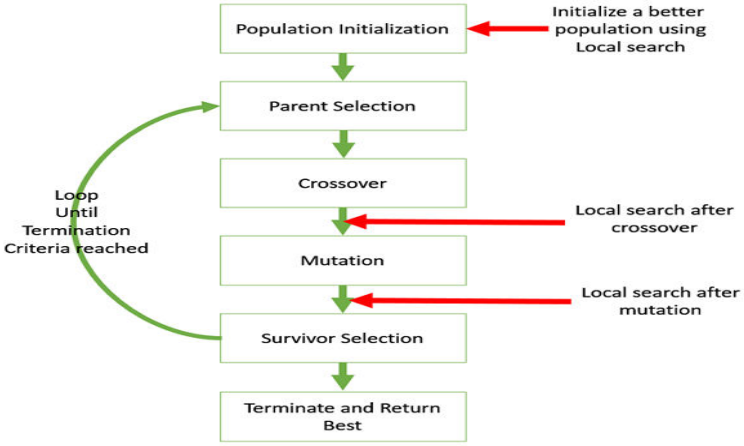
Reduce Crowding:

- When a highly fit chromosome reproduces a lot, the population becomes crowded, and In a few years, the whole nation would comprise equal fitness alternatives. This has the effect of reducing diversity, which is a crucial part of a GA's success. There are a variety of approaches to reducing crowding. There are a few of them.
- Diversity is introduced by mutation**.**
- Used choices based on rank and tournament, for people with equal fitness, with a higher selection pressure than fitness proportionate selection.

Fitness Sharing: When a population already includes people with similar fitness levels, an individual's fitness is decreased.

Hybridize GA with Local Search:

Local search is the process of testing solutions in the vicinity of a given solution for improved objective values. Hybridizing the GA with local search can be beneficial at times. The diagram below depicts the different locations in a GA where local search can be used.



Variation of Parameters and Techniques:

There is no such thing as a "one-size-fits-everything" genetic algorithm or a magic formula that solves all problems. Even after the initial GA is complete, it takes a significant amount of time and effort to experiment with parameters such as population size, mutation, and crossover likelihood to find the ones that best fit the problem.

Conclusions:

In this Paper different functions and operators are explained in detailed. The concept of selection ,mutation, crossover, fitness function and various Models are explained in detail. The richness of Genetic Algorithms and the various parallel models is one of the most striking features of this kind of Computing Technique. Minor changes to the algorithm will often result in unpredictable emergent behaviour. Our understanding of genetic algorithms has also improved due to recent scientific advances, allowing us to use more sophisticated analytical methods. Genetic Algorithm is robust, efficient, easily available and, thus, a powerful tool for optimization.

References:

1. Meyer, L., Feng, X.: A fuzzy stop criterion for genetic algorithms using performance estimation. In: Proceedings of the Third IEEE Conference on Fuzzy Systems. (1994) 1990–1995
2. Howell, M., Gordon, T., Brandao, F.: Genetic learning automata for function optimization. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32 (2002) 804–815
3. Poli, R.: Exact schema theorem and effective fitness for GP with one-point crossover. In Whitley, L.D., Goldberg, D.E., Cant´u-Paz, E., Spector, L., Parmee, I.C., Beyer, H.G., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2000) 469–476
4. Azadivar, F. and Tompkins, G., Simulation optimization with qualitative variables and structural model changes: a genetic algorithm approach. *Euro. J. Opt. Res.*, 1999, **113**(1), 169–183.
5. Chan, K.C., and Tansri, H., A study of genetic crossover operations on the facilities layout problem. *Computers & Ind. Eng.*, 1994, **26**(3), 537–550.
6. Chen, C.L., Vempati, V.S. and Aljaber, N., An application of genetic algorithms for flow shop problems. *Euro. J. Opt. Res.*, 1995, **80**(2), 389–397.
7. De Jong, K.A.: Genetic algorithms are NOT function optimizers. In Whitley, L.D., ed.: Proceedings of the Second Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann (1993) 5–17
8. B¨ack, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: Comments on the history and current state. IEEE Transactions on Evolutionary Computation 1 (1997) 3–17
9. Burkowski FJ (1999) Shuffle crossover and mutual information. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 1999, pp. 1574–1580
10. Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Systems 9:115–148

Websites:

1. https://www.sciencedirect.com/topics/engineering/genetic-algorithm
2. https://www.geeksforgeeks.org/genetic-algorithms/
3. https://ocw.mit.edu/courses/institute-for-data-systems-and-society/ids-338j-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/MITESD_77S10_lec11.pdf

4. http://pages.cs.wisc.edu/~dyer/cs540/notes/ga.html
5. https://www.tutorialspoint.com/genetic_algorithms/index.htm
6. Darrell Whitley, Computer Science Department Colorado State University, Fort Collins CO
7. https://www.cs.jhu.edu/~ayuille/courses/Stat202C-Spring10/ga_tutorial.pdf
8. https://www.researchgate.net/post/How_can_I_decide_the_stopping_criteria_in_Genetic_Algorithm
9. https://www.cs.rochester.edu/u/nelson/courses/csc_173/genetic-algs/algorithm.html
10.