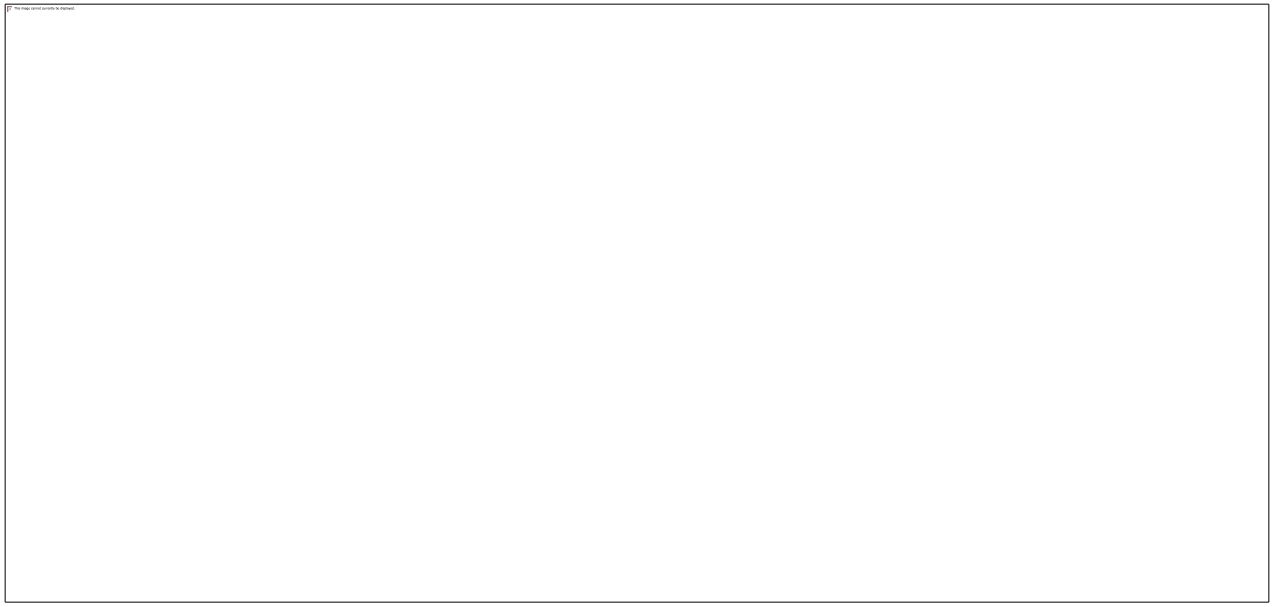
Evolutionary Algorithms

Vijay Kumar Gupta
vkgupta@iiitdmj.ac.in

Evolution

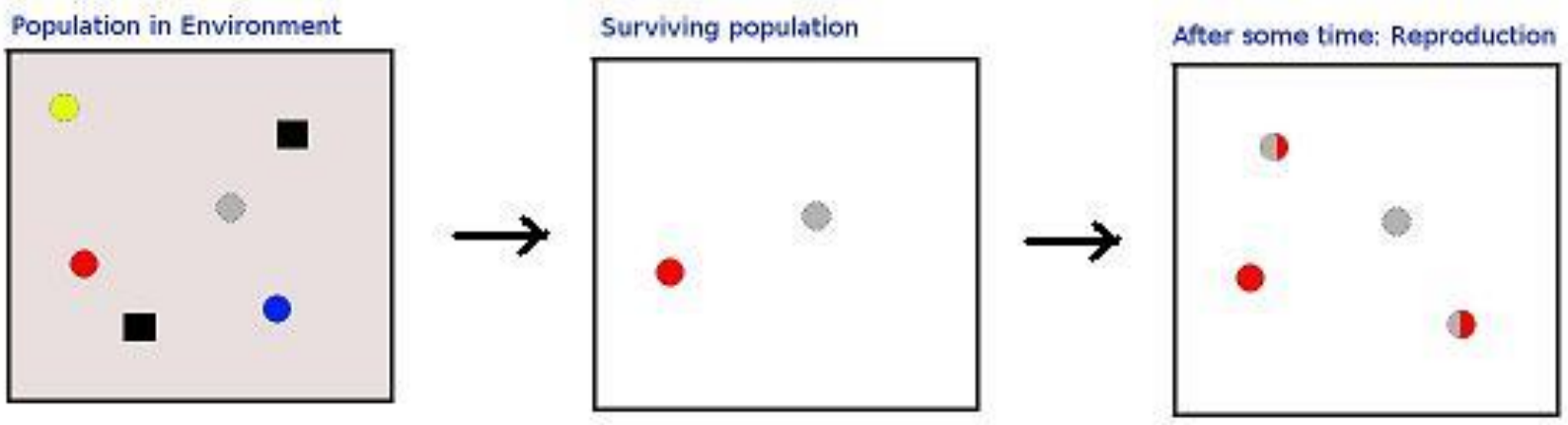
- Evolution is the change in the inherited traits of a population from one generation to the next.



- Natural selection leading to better and better species

Evolution – Fundamental Laws

- Survival of the fittest.
- Change in species is due to change in genes over reproduction or/and due to mutation.



An Example showing the concept of survival of the fittest and reproduction over generations.

What is Evolutionary Computation?

A technique borrowed from the theory of biological evolution that is used to create optimization procedures or methodologies, usually implemented on computers, that are used to solve problems.

Natural Selection

- Limited number of resources
- Competition results in struggle for existence
- Success depends on fitness --
    - fitness of an individual: how well-adapted an individual is to their environment. This is determined by their genes (blueprints for their physical and other characteristics).
- Successful individuals are able to reproduce and pass on their genes

When changes occur ...

- Previously “fit” (well-adapted) individuals will no longer be best-suited for their environment

- Some members of the population will have genes that confer different characteristics than “the norm”. Some of these characteristics can make them more “fit” in the changing environment.

Genetic Change in Individuals

- Mutation in genes
  - may be due to various sources (e.g. UV rays, chemicals, etc.)

  Start:

  1001001001001001001001

Location of Mutation

After Mutation:
1001000001001001001001

Genetic Change in Individuals

- Recombination (Crossover)
  - occurs during reproduction -- sections of genetic material exchanged between two chromosomes

Recombination (Crossover)

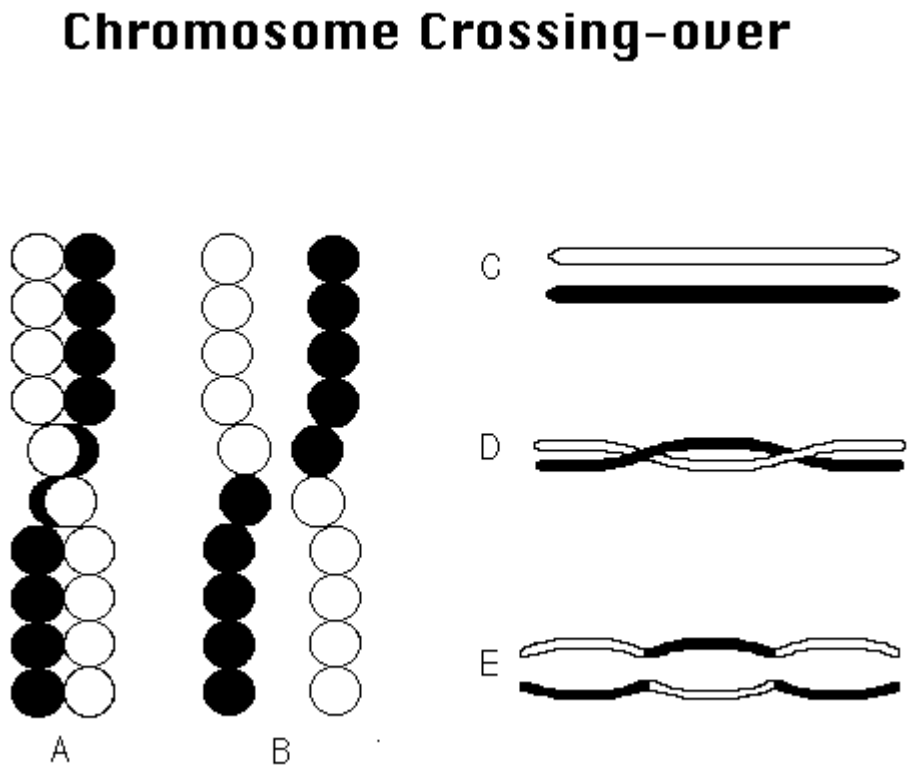


Image from http://esg-www.mit.edu:8001/bio/mg/meiosis.html

The Nature of Computational Problems

- Require search through many possibilities to find a solution
    - (e.g. search through sets of rules for one set that best predicts the ups and downs of the financial markets)
  - Search space too big -- search won't return within our lifetimes
- Require algorithm to be adaptive or to construct original solution
    - (e.g. interfaces that must adapt to idiosyncrasies of different users)

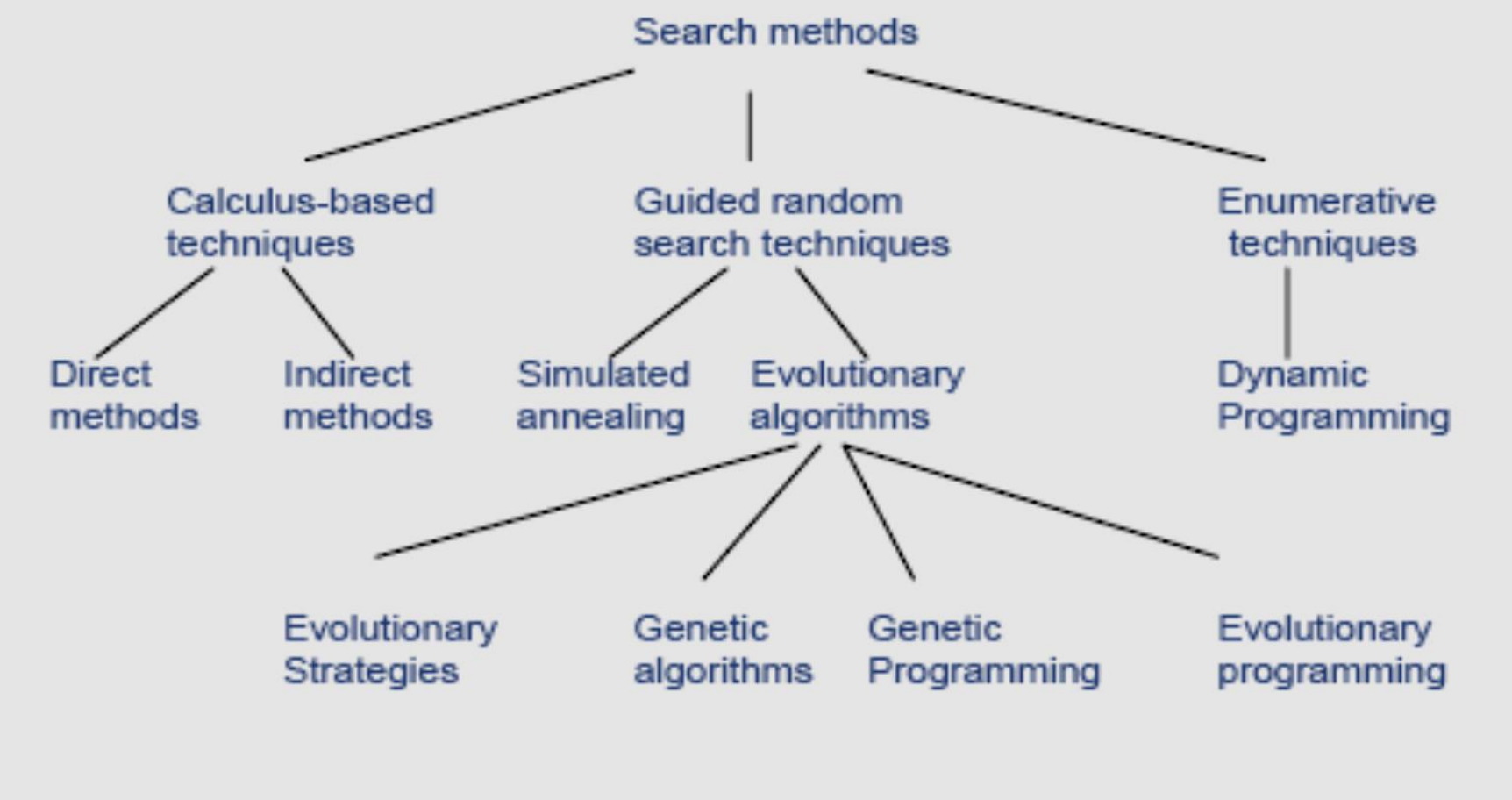
Why Evolution Proves to be a Good Model for Solving these Types of Problems

- Evolution is a method of searching for an (almost) optimal solution
  - Possibilities -- all individuals
  - Best solution -- the most “fit” or well-adapted individual
- Evolution is a parallel process
  - Testing and changing of numerous species and individuals occur at the same time (or, in parallel)
- Evolution can be seen as a method that designs new (original) solutions to a changing environment

Evolutionary Computing

- Genetic Algorithms
  - invented by John Holland (University of Michigan) in the 1960's
- Evolution Strategies
  - invented by Ingo Rechenberg (Technical University Berlin) in the 1960's
- Started out as individual developments, but converged in the later years

Search Methods



Genetic Algorithm (GA)

- Search-based optimization technique based on the principles of **Genetics and Natural Selection**.
- It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve.
- It is frequently used to solve optimization problems, in research, and in machine learning.
- GAs are a subset of a much larger branch of computation known as **Evolutionary Computation**.
- GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success.

- In GAs, we have a **pool or a population of possible solutions** to the given problem.

- These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations.

- Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more “fitter” individuals. This is in line with the Darwinian Theory of “Survival of the Fittest”.

- In this way we keep “evolving” better individuals or solutions over generations, till we reach a stopping criterion.

- Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well.

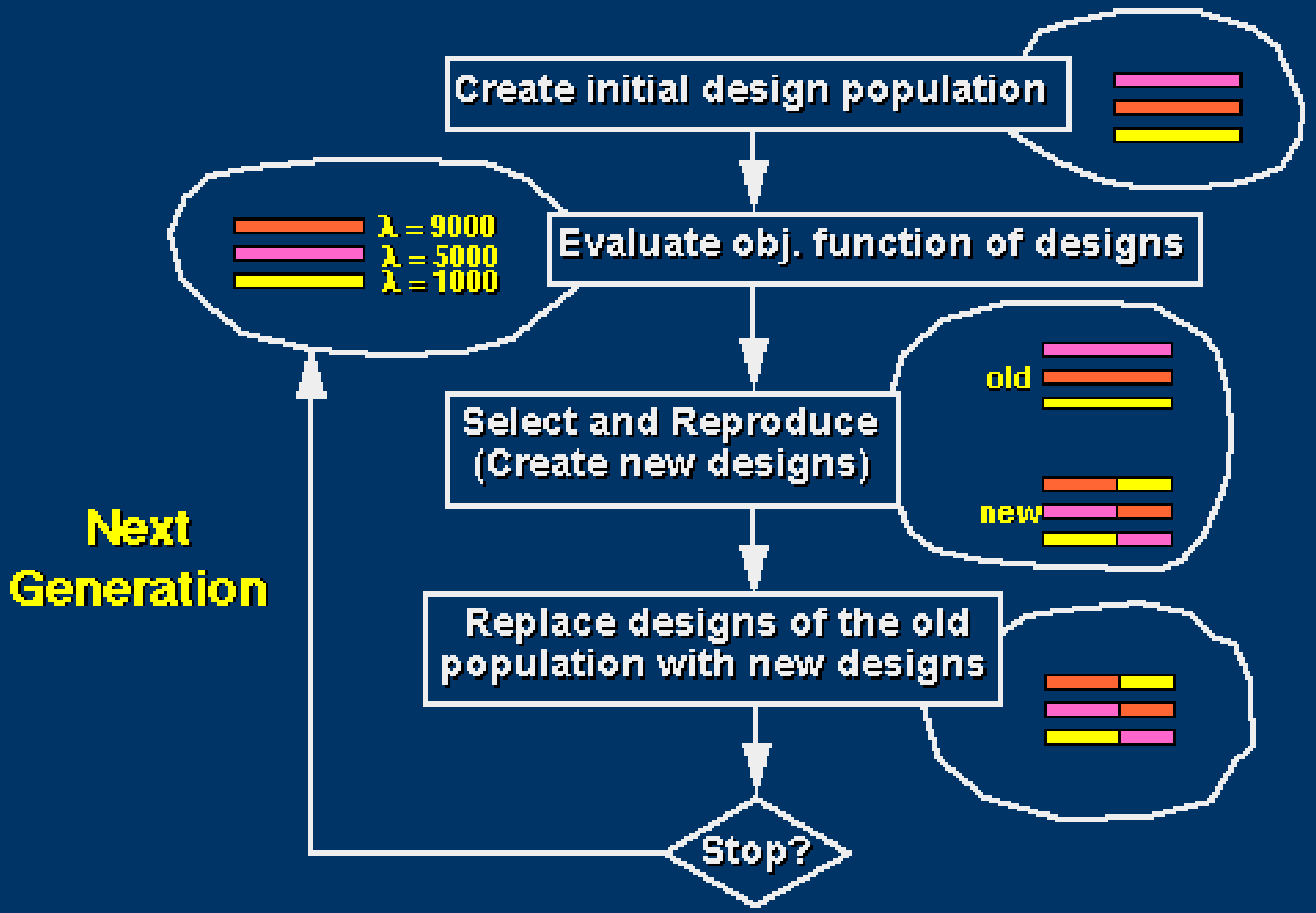
Advantages of GAs

- Does not require any derivative information (which may not be available for many real-world problems).
- Is faster and more efficient as compared to the traditional methods.
- Has very good parallel capabilities.
- Optimizes both continuous and discrete functions and also multi-objective problems.
- Provides a list of “good” solutions and not just a single solution.
- Always gets an answer to the problem, which gets better over the time.
- Useful when the search space is very large and there are a large number of parameters involved.

Limitations of GAs

- GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.
- If not implemented properly, the GA may not converge to the optimal solution.

GA Flowchart

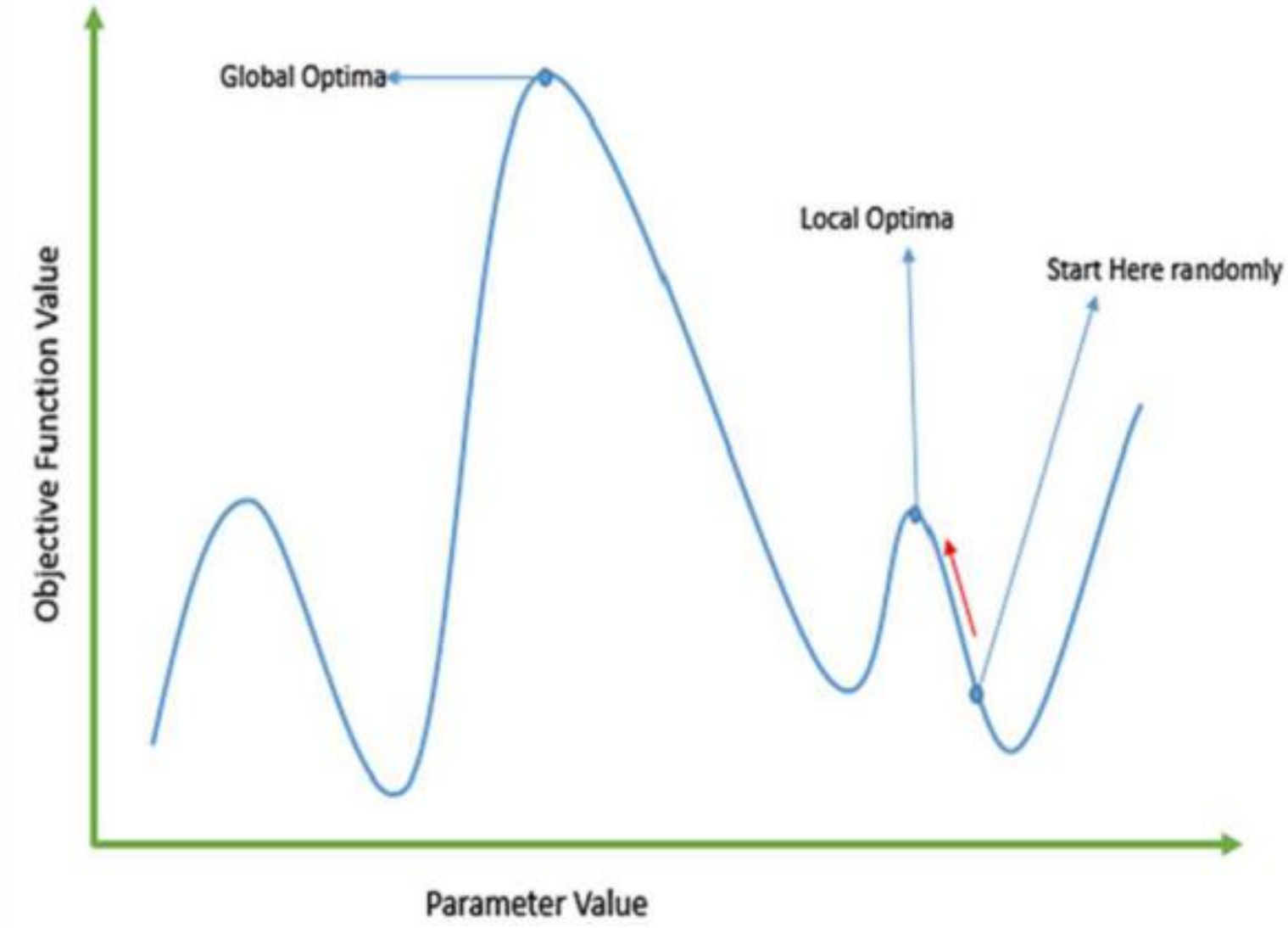


Why GA

- There is a large set of problems, which are **NP-Hard**. What this essentially means is that, even the most powerful computing systems take a very long time (even years!) to solve that problem. In such a scenario, GAs prove to be an efficient tool to provide **usable near-optimal solutions** in a short amount of time.

- Traditional calculus based methods work by starting at a random point and by moving in the direction of the gradient, till we reach the top of the hill. This technique is efficient and works very well for unimodal objective functions like the cost function in linear regression.

- But, in most real-world situations, we have a very complex problem called as landscapes, which are multimodal in nature. For such problems gradient methods does not provide solution as they get stuck at the local optima as shown in the figure.

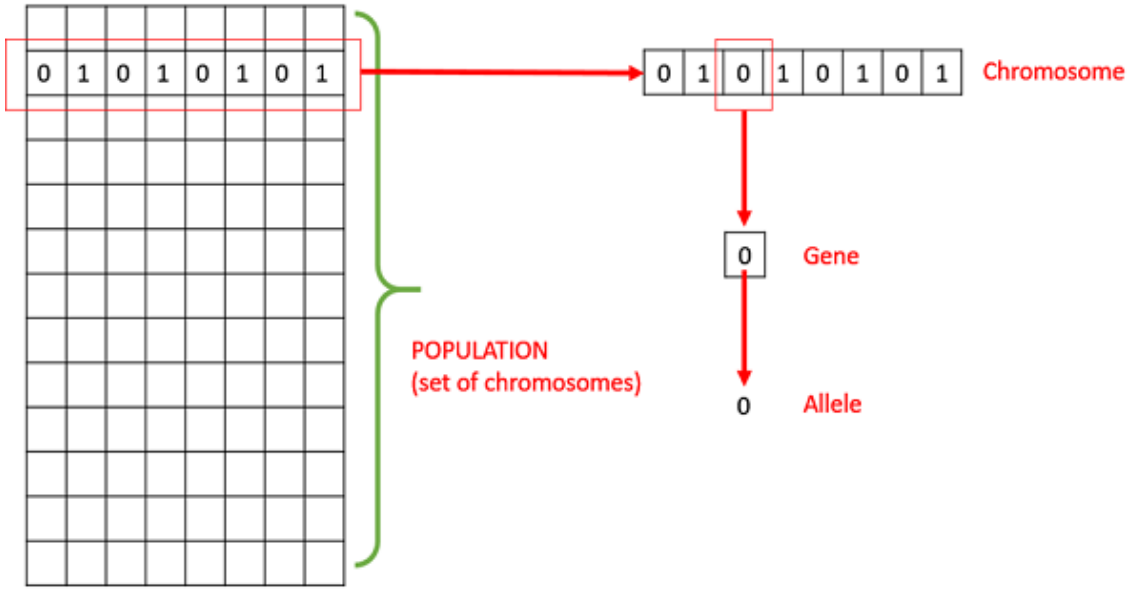

Global Optima
Local Optima
Start Here randomly
Objective Function Value
Parameter Value

- Getting a Good Solution Fast

- Some difficult problems like the Travelling Salesperson Problem (TSP), have real-world applications like path finding. Suppose, you are moving on a road and using your GPS Navigation system. It takes a few minutes (or even a few hours) to compute the “optimal” path from the source to destination. Delay in such real world applications is not acceptable and therefore a “good-enough” solution, which is delivered “fast” is what is required.

Basic Terminology

Population – It is a subset of all the possible (encoded) solutions to the given problem.
Chromosomes – A chromosome is one such solution to the given problem.
Gene – A gene is one element position of a chromosome.
Allele – It is the value a gene takes for a particular chromosome.



Genotype − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.

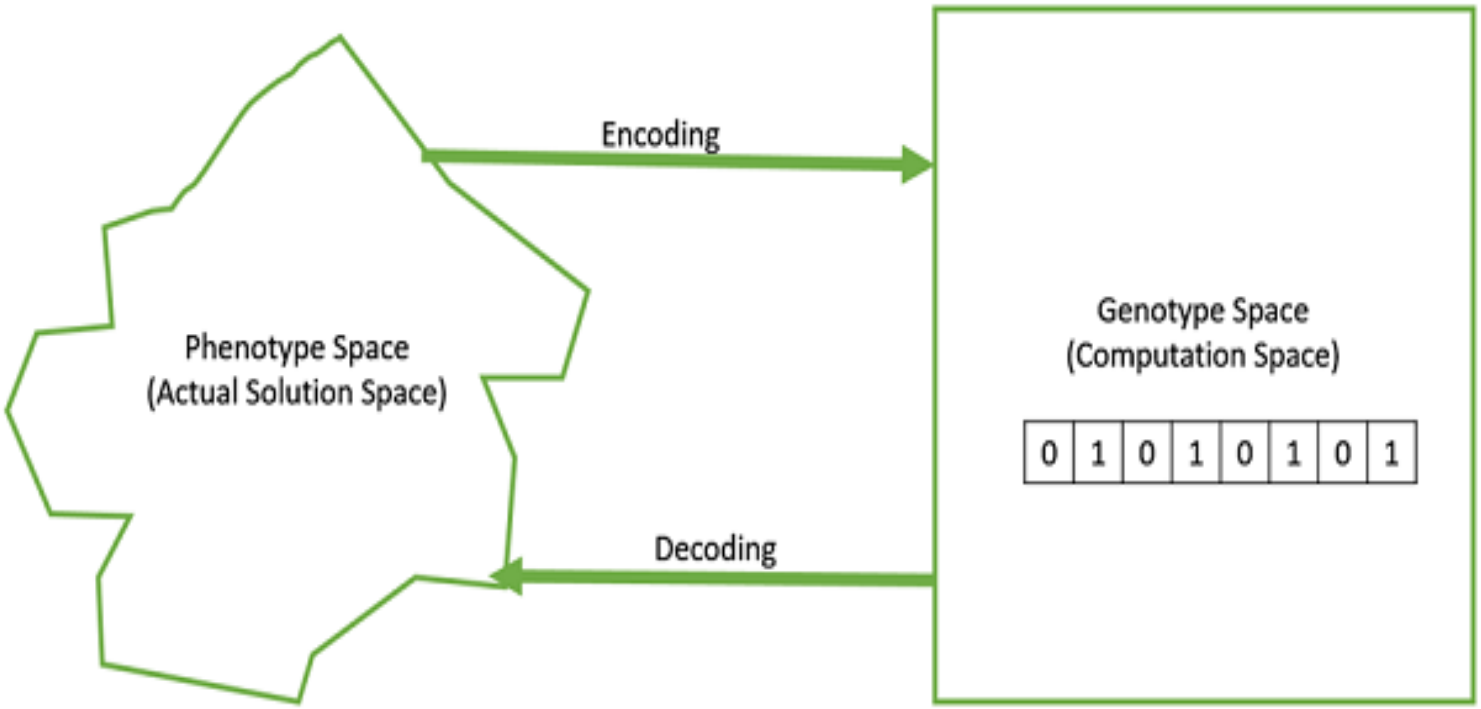
Phenotype − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

Decoding and Encoding − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

Genotype space can be represented as a binary string of length n (where n is the number of items).

A **0 at position x** represents that **x^{th}** item is picked while a 1 represents the reverse.

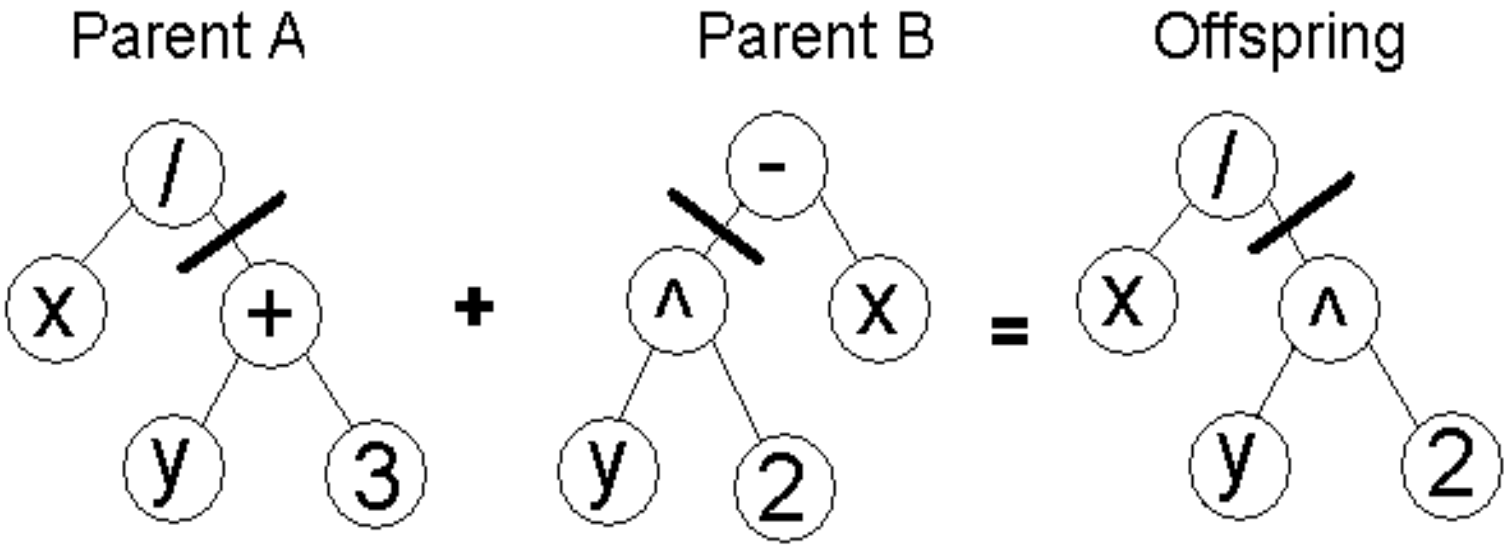
This is a case where genotype and phenotype spaces are different.



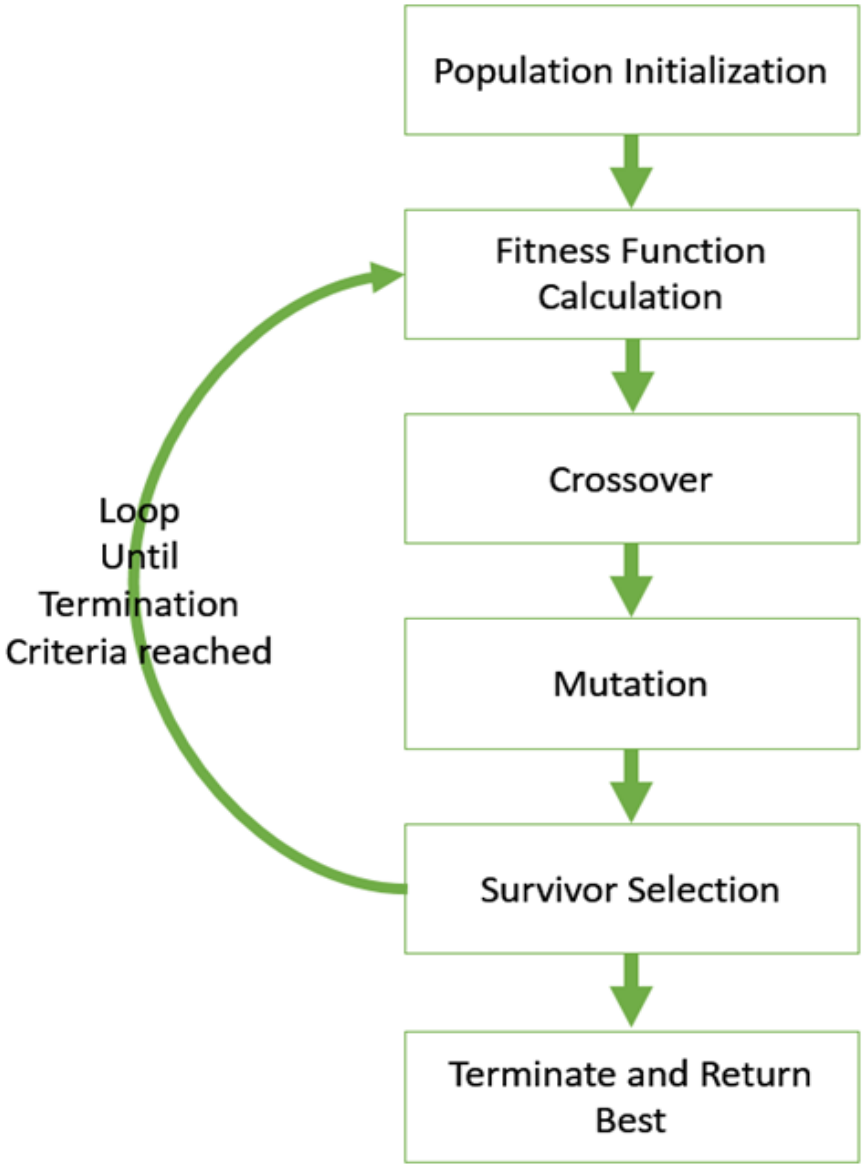
Fitness Function − A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.

Genetic Operators − These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

A Combination Operator for Expressions



Basic Structure



Individual Encoding/ Representation

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Representation

Representation is very important in GA. It has been observed that improper representation can lead to poor performance of the GA.

Binary Representation

- This is one of the simplest and most widely used representation in GAs. In this type of representation the genotype consists of bit strings.

- For Boolean decision variables – yes or no, the binary representation is natural.
- For other problems, specifically those dealing with numbers, we can represent the numbers with their binary representation. The problem with this is that different bits have different significance and therefore mutation and crossover operators can have undesired consequences.
- This can be resolved to some extent by using **Gray Coding,** as a change in one bit does not have a massive effect on the solution.

Real Valued Representation

- For problems where we want to define the genes using continuous rather than discrete variables, the real valued representation is the most natural. The precision of these real valued or floating point numbers is however limited to the computer.

Integer Representation

- For discrete valued genes, we cannot always limit the solution space to binary ‘yes’ or ‘no’. For example, if we want to encode the four distances – North, South, East and West, we can encode them as **{0,1,2,3}**. In such cases, integer representation is desirable.

Permutation Representation

- In many problems, the solution is represented by an order of elements. In such cases permutation representation is the most suited.

- A classic example of this representation is the travelling salesman problem (TSP). In this the salesman has to take a tour of all the cities, visiting each city exactly once and come back to the starting city. The total distance of the tour has to be minimized. The solution to this TSP is naturally an ordering or permutation of all the cities and therefore using a permutation representation makes sense for this problem.

Population

Population is a subset of solutions in the current generation. It can also be defined as a set of chromosomes.

The diversity of the population should be maintained otherwise it might lead to premature convergence.

The population size should not be kept very large as it can cause a GA to slow down, while a smaller population might not be enough for a good mating pool. Therefore, an optimal population size needs to be decided by trial and error.

The population is usually defined as a two dimensional array of – **size population, size x, chromosome size**.

Population Initialization

- There are two primary methods to initialize a population in a GA. **Random Initialization** – Populate the initial population with completely random solutions.
- **Heuristic initialization** – Populate the initial population using a known heuristic for the problem.

- It has been observed that the entire population should not be initialized using a heuristic, as it can result in the population having similar solutions and very little diversity.

- It has been experimentally observed that the random solutions are the ones to drive the population to optimality.

- It has also been observed that heuristic initialization in some cases, only effects the initial fitness of the population, but in the end, it is the diversity of the solutions which lead to optimality.

Population Models

There are two population models widely in use −

- **Steady State**
- In steady state GA, we generate one or two off-springs in each iteration and they replace one or two individuals from the population. A steady state GA is also known as **Incremental GA**.

- **Generational**
- In a generational model, we generate ‘n’ off-springs, where n is the population size, and the entire population is replaced by the new one at the end of the iteration.

Fitness Function

- The fitness function simply defined is a function which takes a **candidate solution to the problem as input and produces as output** how “fit” our how “good” the solution is with respect to the problem in consideration.

A fitness function should possess the following characteristics −

- The fitness function should be sufficiently fast to compute.
- It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.
- In some cases, calculating the fitness function directly might not be possible due to the inherent complexities of the problem at hand. In such cases, we do fitness approximation to suit our needs.

Selection

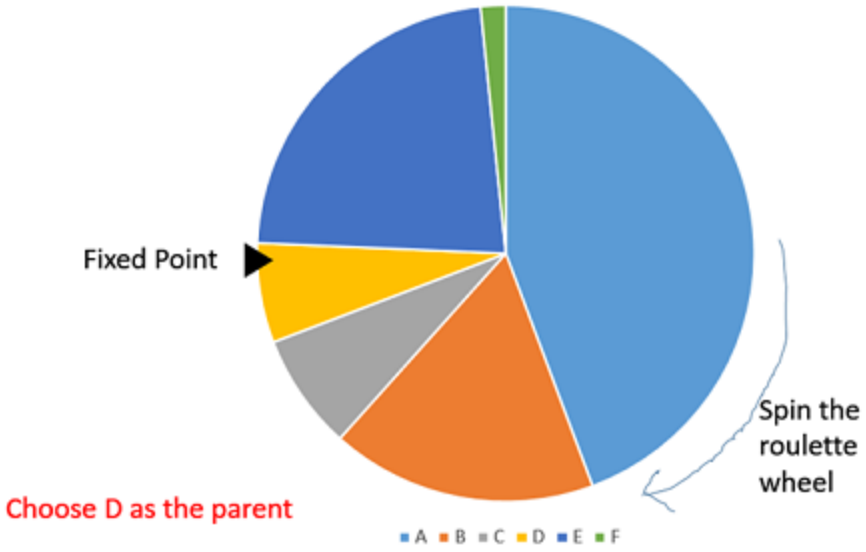
- Parent Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.
- Parent selection is very crucial to the convergence rate of the GA as good parents drive individuals to a better and fitter solutions.
- However, care should be taken to prevent one extremely fit solution from taking over the entire population in a few generations, as this leads to the solutions being close to one another in the solution space thereby leading to a loss of diversity.
- **Maintaining good diversity** in the population is extremely crucial for the success of a GA.
- This taking up of the entire population by one extremely fit solution is known as **premature convergence** and is an undesirable condition in a GA.

Fitness Proportionate Selection

- Fitness Proportionate Selection is one of the most popular ways of parent selection.
- In this every individual can become a parent with a probability which is proportional to its fitness. Therefore, fitter individuals have a higher chance of mating and propagating their features to the next generation.
- Therefore, such a selection strategy applies a selection pressure to the more fit individuals in the population, evolving better individuals over time.
- Consider a circular wheel. The wheel is divided into **n pies**, where n is the number of individuals in the population. Each individual gets a portion of the circle which is proportional to its fitness value.

Roulette Wheel Selection

- In a roulette wheel selection, the circular wheel is divided as described before. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent.
- For the second parent, the same process is repeated.



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

- It is clear that a fitter individual has a greater area on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated.
- Therefore, the probability of choosing an individual depends directly on its fitness.

Implementation −

- Calculate S = the sum of a fitness.
- Generate a random number between 0 and S.
- Starting from the top of the population, keep adding the fitness to the partial sum P, till P<S.
- The individual for which P exceeds S is the chosen individual.

Roulette wheel selection

Consider a population containing four strings shown

Candidate	Fitness value	Percentage of total fitness
1011 0110 1101 1001	109	28.09
0101 0011 1110 1101	76	19.59
0001 0001 1111 1011	50	12.89
1011 1111 1011 1100	153	39.43
Total	388	100

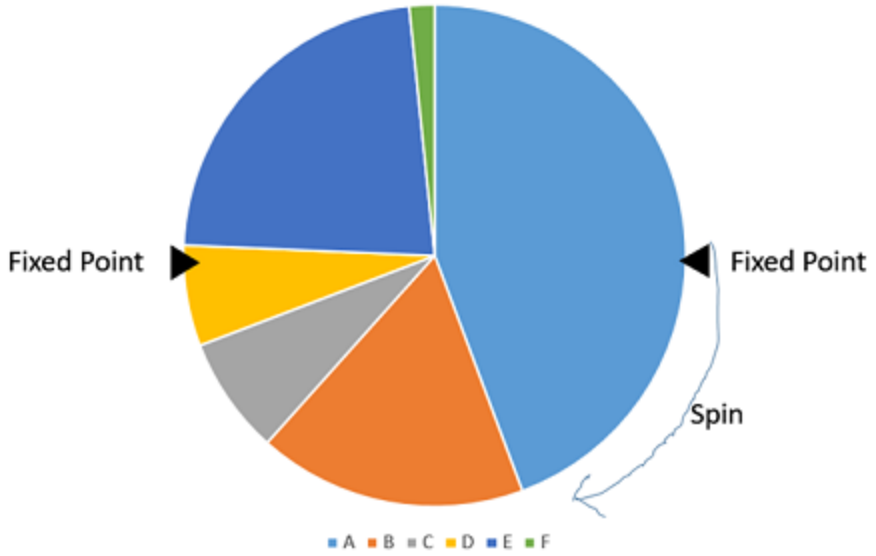
- Each string is formed by concatenating four substrings representing variables a, b, c and d. Length of each string is taken as four bits

Parent Selection: Roulette wheel selection

- These probabilities are represented on a pie chart
- Then four numbers are randomly generated between 1 and 100
- The likeliness of these numbers falling in the region of candidate 2 might be once, whereas for candidate 4 it might be twice and candidate 1 more than once and for candidate 3 it may not fall at all
- Thus, the strings are chosen to form the parents of the next generation
- The main disadvantage of this method is when the fitnesses differ very much
- For example, if the best chromosome fitness is 90% of the entire roulette wheel then the other chromosomes will have very few chances to be selected

Stochastic Universal Sampling (SUS)

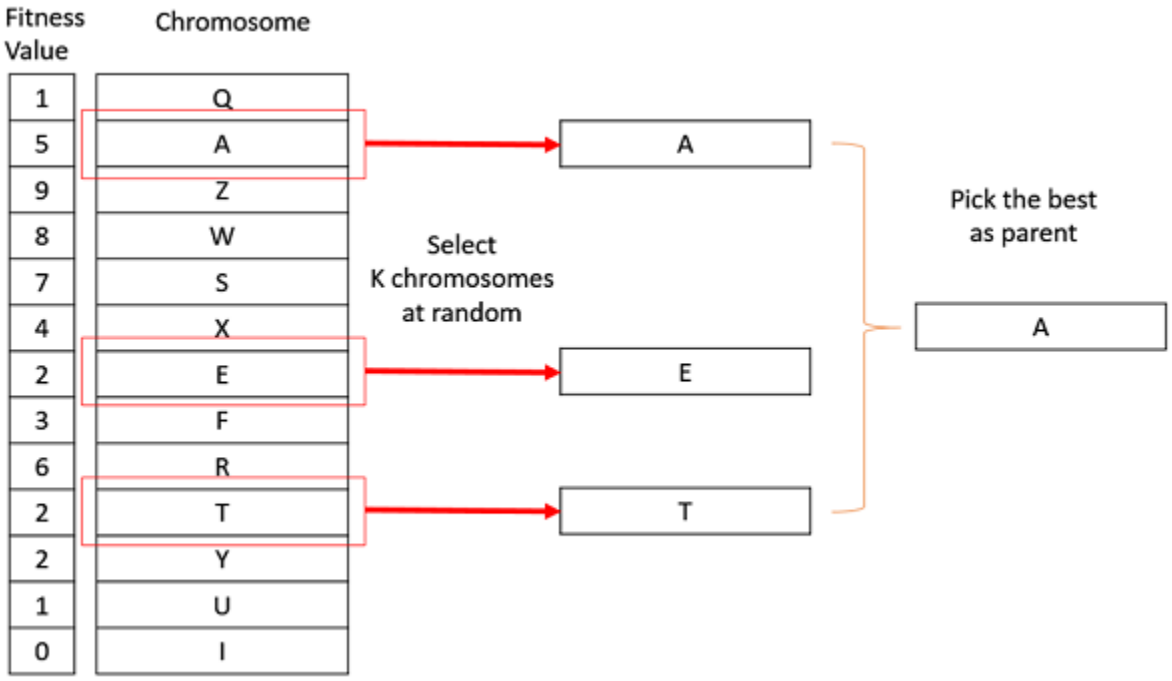
- Stochastic Universal Sampling is quite similar to Roulette wheel selection, however instead of having just one fixed point, we have multiple fixed points as shown in the following image.
- Therefore, all the parents are chosen in just one spin of the wheel. Also, such a setup encourages the highly fit individuals to be chosen at least once.
- It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.



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

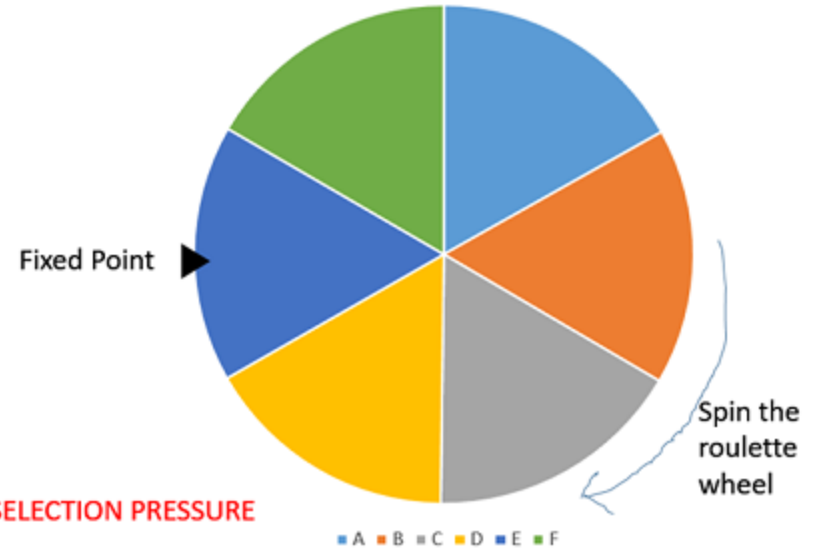
Tournament Selection

- In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent.
- The same process is repeated for selecting the next parent.
- Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.



Rank Selection

- Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run).
- This leads to each individual having an almost equal share of the pie and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent.
- This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such



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

- The concept of a fitness value is removed while selecting a parent. However, every individual in the population is ranked according to their fitness.
- The selection of the parents depends on the rank of each individual and not the fitness.
- The higher ranked individuals are preferred more than the lower ranked ones.

Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

Random Selection

- In this strategy parents are randomly selected from the existing population.
- There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

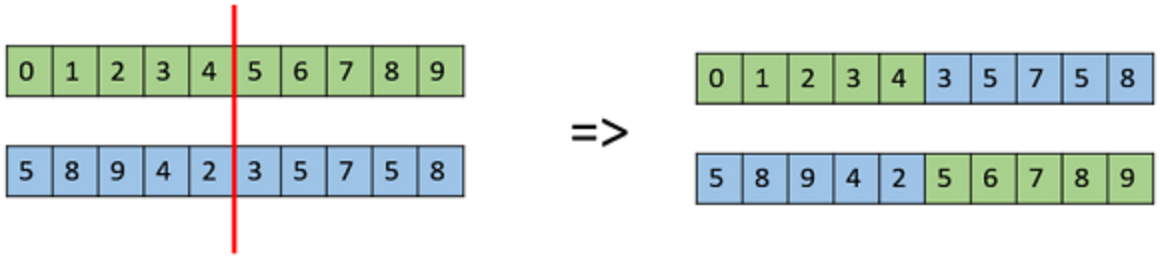
CROSSOVER

- The crossover operator is analogous to reproduction and biological crossover.
- In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents.
- Crossover is usually applied in a GA with a high probability

Crossover Operators

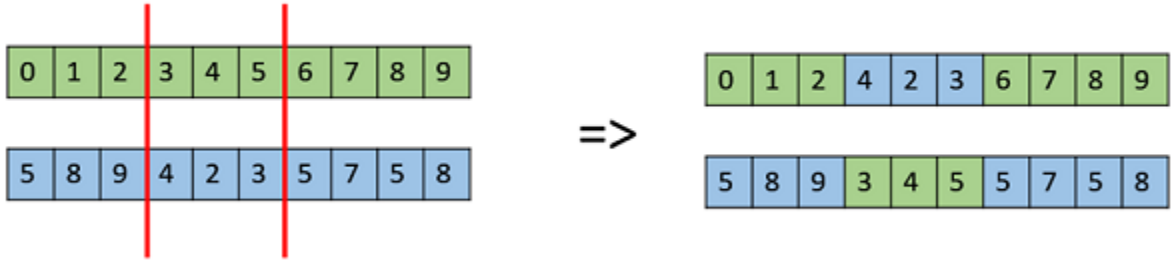
One Point Crossover

- In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.



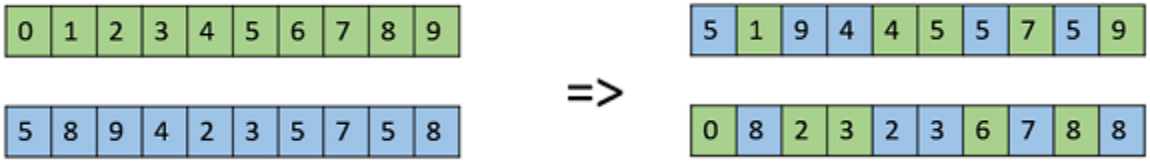
Multi Point Crossover

- Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



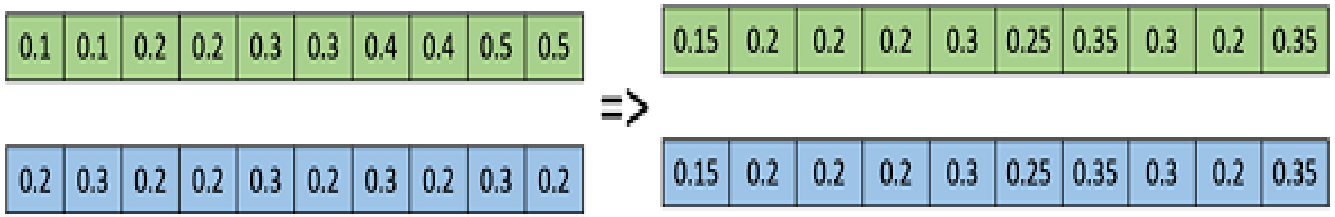
Uniform Crossover

- In a uniform crossover, the chromosome are not divided into segments, rather treated separately.
- In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring.
- We can also bias the coin to one parent, to have more genetic material in the child from that parent.



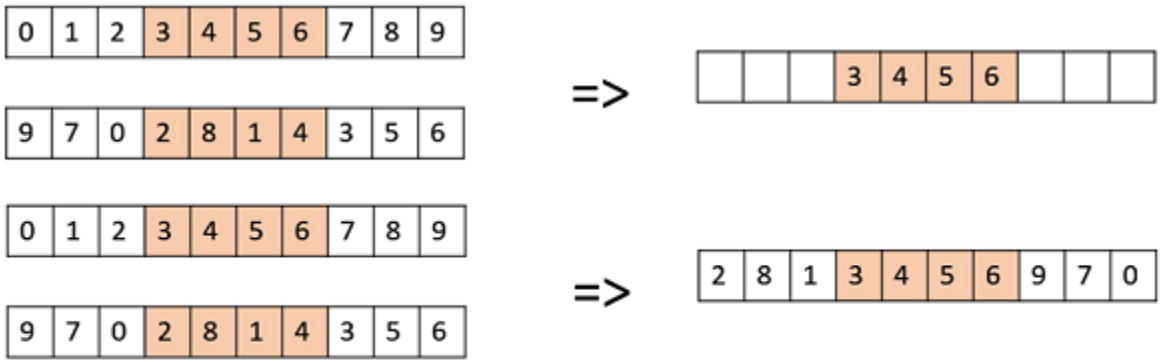
Whole Arithmetic Recombination

- This is commonly used for integer representations and works by taking the weighted average of the two parents by using the following formulae −
- Child1 = α.x + (1-α).y
- Child2 = (1-α).x + α.y
- Obviously, if α = 0.5, then both the children will be identical as shown in the following image.



Davis' Order Crossover (OX1)

- OX1 is used for permutation based crossovers with the intention of transmitting information about relative ordering to the off-springs.
- It works as follows −
  - Create two random crossover points in the parent and copy the segment between them from the first parent to the first offspring.
  - Now, starting from the second crossover point in the second parent, copy the remaining unused numbers from the second parent to the first child, wrapping around the list.
  - Repeat for the second child with the parent's role reversed.



Repeat the same procedure to get the second child

There exist a lot of other crossovers like

- Partially Mapped Crossover (PMX),
- Order based crossover (OX2),
- Shuffle Crossover,
- Ring Crossover, etc.

MUTATION

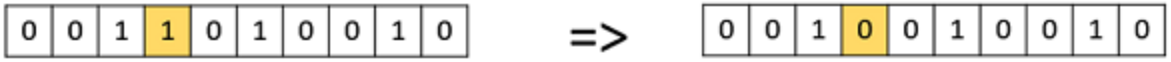
- Mutation may be defined as a small random tweak in the chromosome, to get a new solution.
- It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – $\boldsymbol{p_m}$.
- If the probability is very high, the GA gets reduced to a random search.
- Mutation is the part of the GA which is related to the "exploration" of the search space. It has been observed that mutation is essential to the convergence of the GA while crossover is not.

Mutation Operators

In this section, we describe some of the most commonly used mutation operators. Like the crossover operators, this is not an exhaustive list and the GA designer might find a combination of these approaches or a problem-specific mutation operator more useful.

Bit Flip Mutation

In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.

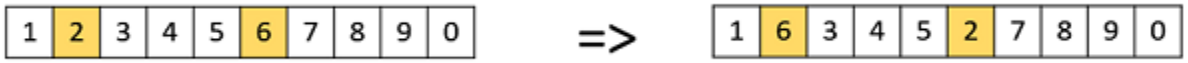


Random Resetting

- Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

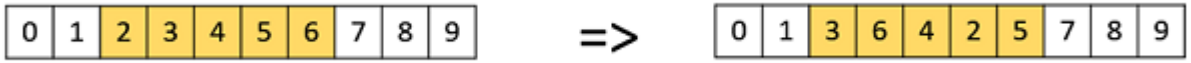
Swap Mutation

- In swap mutation, we select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.



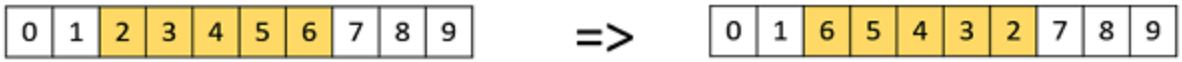
Scramble Mutation

- Scramble mutation is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



Inversion Mutation

- In inversion mutation, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.

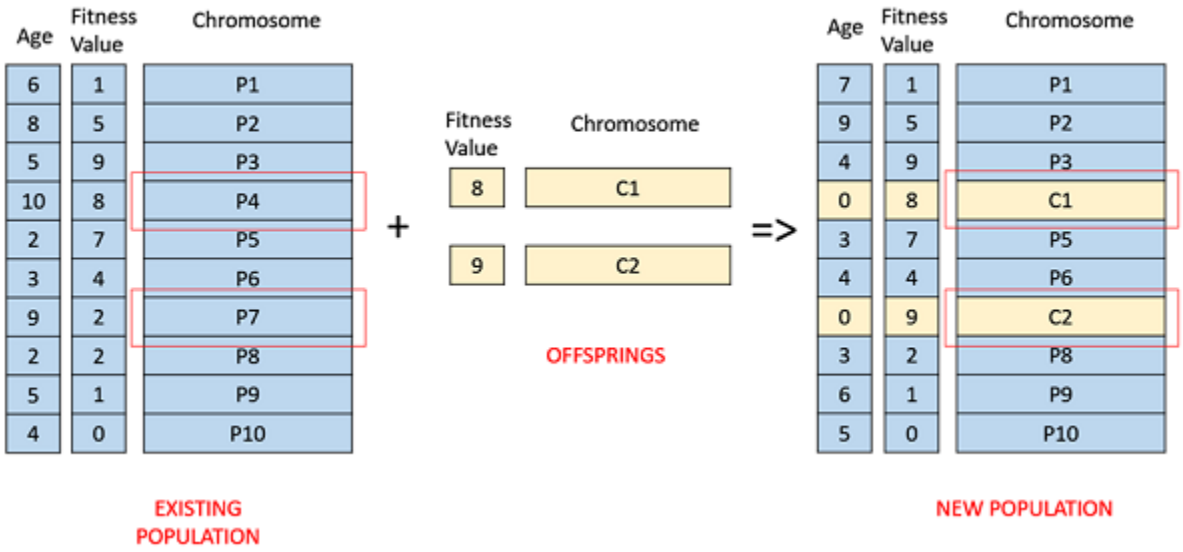


SURVIVOR SELECTION

- The Survivor Selection Policy determines which individuals are to be kicked out and which are to be kept in the next generation.
- It is crucial as it should ensure that the fitter individuals are not kicked out of the population, while at the same time diversity should be maintained in the population.
- Some GAs employ **Elitism**. In simple terms, it means the current fittest member of the population is always propagated to the next generation. Therefore, under no circumstance can the fittest member of the current population be replaced.
- The easiest policy is to kick random members out of the population, but such an approach frequently has convergence issues, therefore the following strategies are widely used.

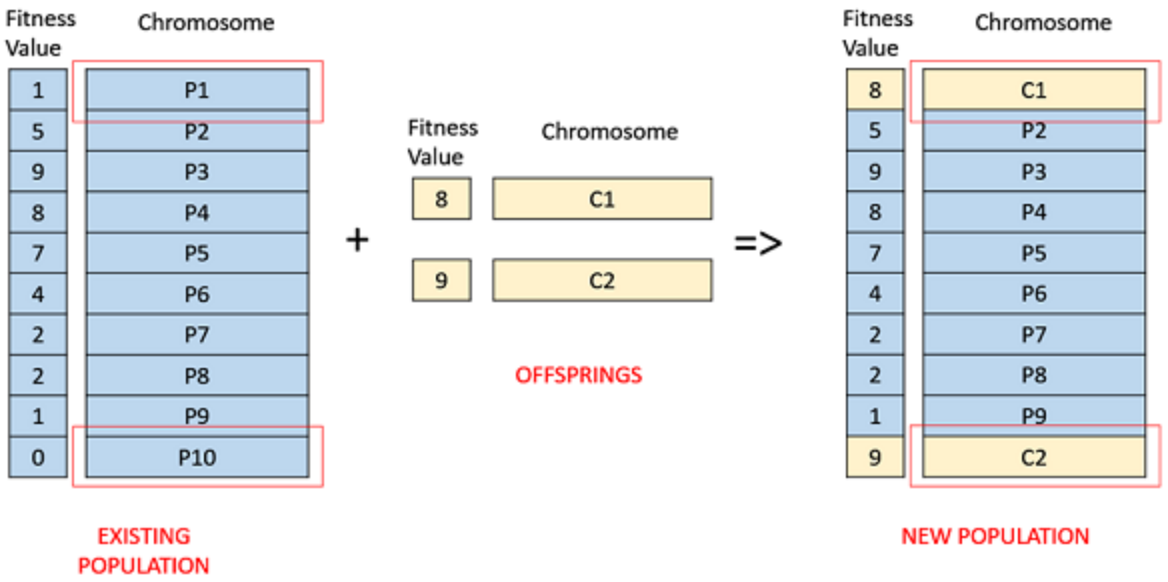
Age Based Selection

- In Age-Based Selection, we don't have a notion of a fitness.
- In this, each individual is allowed in the population for a finite generation where it is allowed to reproduce, after that, it is kicked out of the population no matter how good its fitness is.
- For instance, in the following example, the age is the number of generations for which the individual has been in the population. The oldest members of the population i.e. P4 and P7 are kicked out of the population and the ages of the rest of the members are incremented by one.



Fitness Based Selection

- In this fitness based selection, the children tend to replace the least fit individuals in the population. The selection of the least fit individuals may be done using a variation of any of the selection policies such as tournament selection, fitness proportionate selection, etc.
- For example, in the following image, the children replace the least fit individuals P1 and P10 of the population. It is to be noted that since P1 and P9 have the same fitness value, the decision to remove which individual from the population is arbitrary.



TERMINATION

- The termination condition of a Genetic Algorithm is important in determining when a GA run will end.

- It has been observed that initially, the GA progresses very fast with better solutions coming in every few iterations, but this tends to saturate in the later stages where the improvements are very small.

- We usually want a termination condition such that our solution is close to the optimal, at the end of the run.

Usually, we keep one of the following termination conditions −

- When there has been no improvement in the population for X iterations.
- When we reach an absolute number of generations.
- When the objective function value has reached a certain pre-defined value.
- For example, in a genetic algorithm we keep a counter which keeps track of the generations for which there has been no improvement in the population. Initially, we set this counter to zero. Each time we don't generate off-springs which are better than the individuals in the population, we increment the counter.
- However, if the fitness any of the off-springs is better, then we reset the counter to zero. The algorithm terminates when the counter reaches a predetermined value.
- Like other parameters of a GA, the termination condition is also highly problem specific and the user should try out various options to see what suits his particular problem the best.

ADAPTION

- Other models of lifetime adaptation – **Lamarckian Model** and **Baldwinian Model** also do exist.

- It is to be noted that whichever model is the best, is open for debate and the results obtained by researchers show that the choice of lifetime adaptation is highly problem specific.

- Often, we hybridize a GA with local search – like in Memetic Algorithms.

- In such cases, one might choose do go with either Lamarckian or Baldwinian Model to decide what to do with individuals generated after the local search.

Lamarckian Model

- The Lamarckian Model essentially says that the traits which an individual acquires in his/her lifetime can be passed on to its offspring. It is named after French biologist Jean-Baptiste Lamarck.

- Even though, natural biology has completely disregarded Lamarckism as we all know that only the information in the genotype can be transmitted.

- However, from a computation view point, it has been shown that adopting the Lamarckian model gives good results for some of the problems.

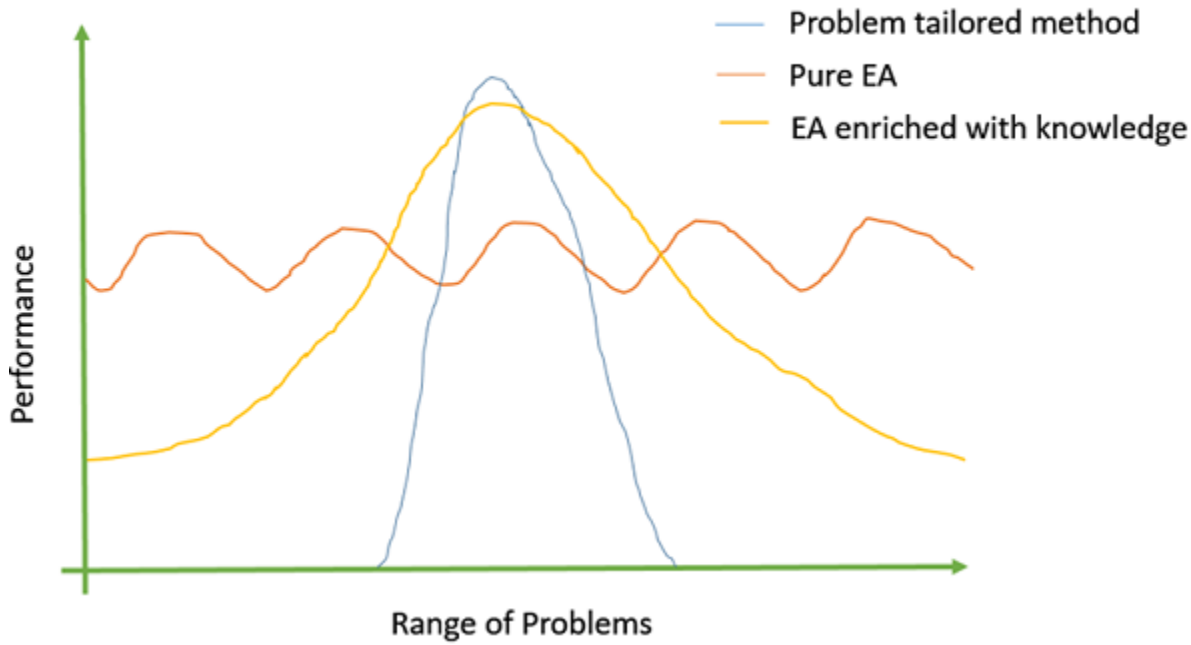
- In the Lamarckian model, a local search operator examines the neighborhood (acquiring new traits), and if a better chromosome is found, it becomes the offspring

Baldwinian Model

- The Baldwinian model is an intermediate idea named after James Mark Baldwin (1896).
- In the Baldwin model, the chromosomes can encode a tendency of learning beneficial behaviors. This means, that unlike the Lamarckian model, we don't transmit the acquired traits to the next generation, and neither do we completely ignore the acquired traits like in the Darwinian Model.
- The Baldwin Model is in the middle of these two extremes, wherein the tendency of an individual to acquire certain traits is encoded rather than the traits themselves.
- In this Baldwinian Model, a local search operator examines the neighborhood (acquiring new traits), and if a better chromosome is found, it only assigns the improved fitness to the chromosome and does not modify the chromosome itself.
- The change in fitness signifies the chromosomes capability to "acquire the trait", even though it is not passed directly to the future generations.

Introduce problem-specific domain knowledge

- It has been observed that the more problem-specific domain knowledge we incorporate into the GA; the better objective values we get.
- Adding problem specific information can be done by either using problem specific crossover or mutation operators, custom representations, etc.
- The following image shows Michalewicz's (1990) view of the EA –

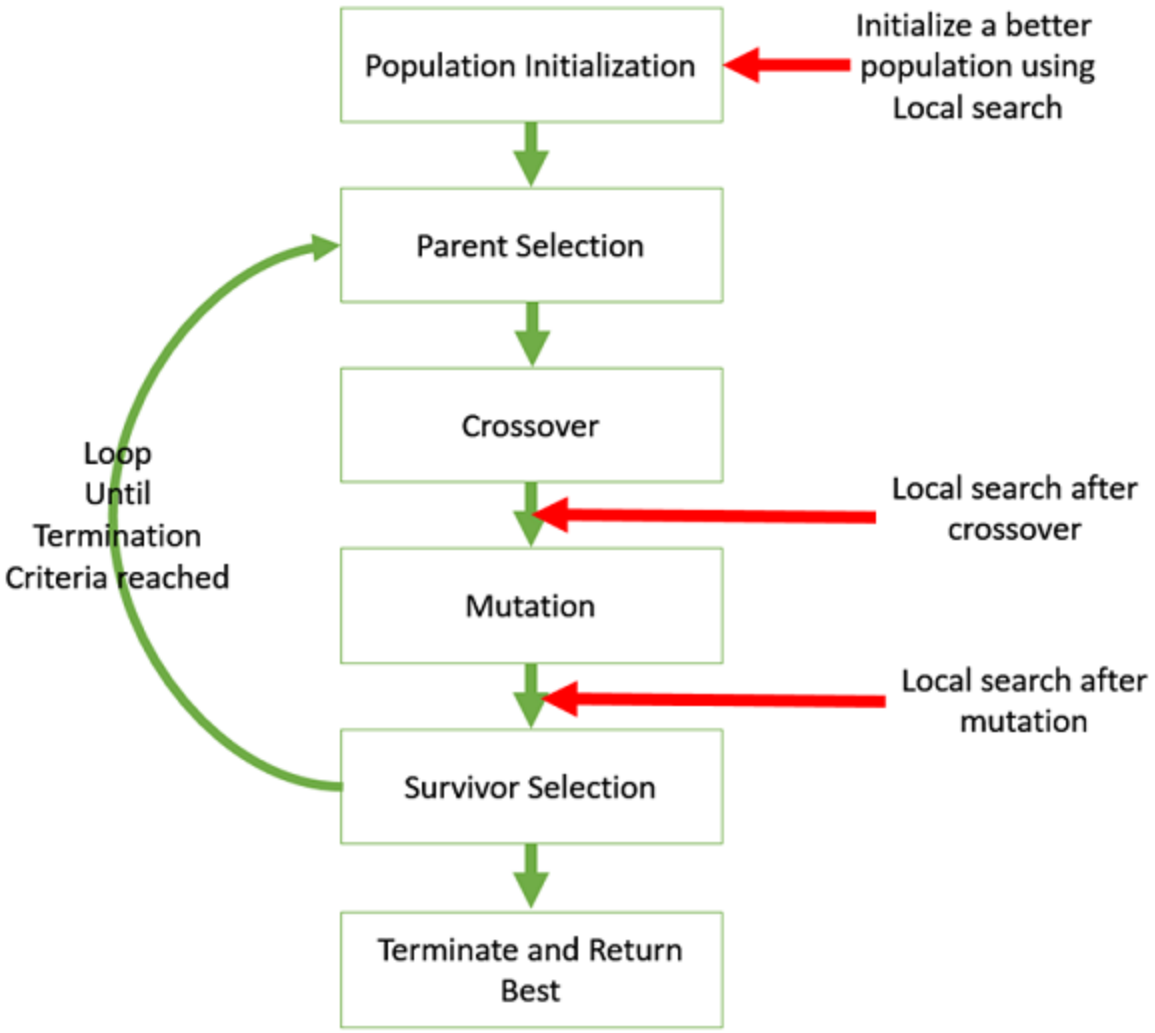


Reduce Crowding

- Crowding happens when a highly fit chromosome gets to reproduce a lot, and in a few generations, the entire population is filled with similar solutions having similar fitness.
- This reduces diversity which is a very crucial element to ensure the success of a GA. There are numerous ways to limit crowding. Some of them are −
  - **Mutation** to introduce diversity.
  - Switching to **rank selection** and **tournament selection** which have more selection pressure than fitness proportionate selection for individuals with similar fitness.
  - **Fitness Sharing** − In this an individual's fitness is reduced if the population already contains similar individuals.

Hybridize GA with Local Search

- Local search refers to checking the solutions in the neighborhood of a given solution to look for better objective values.
- It may be sometimes useful to hybridize the GA with local search.
- The image shows the various places in which local search can be introduced in a GA



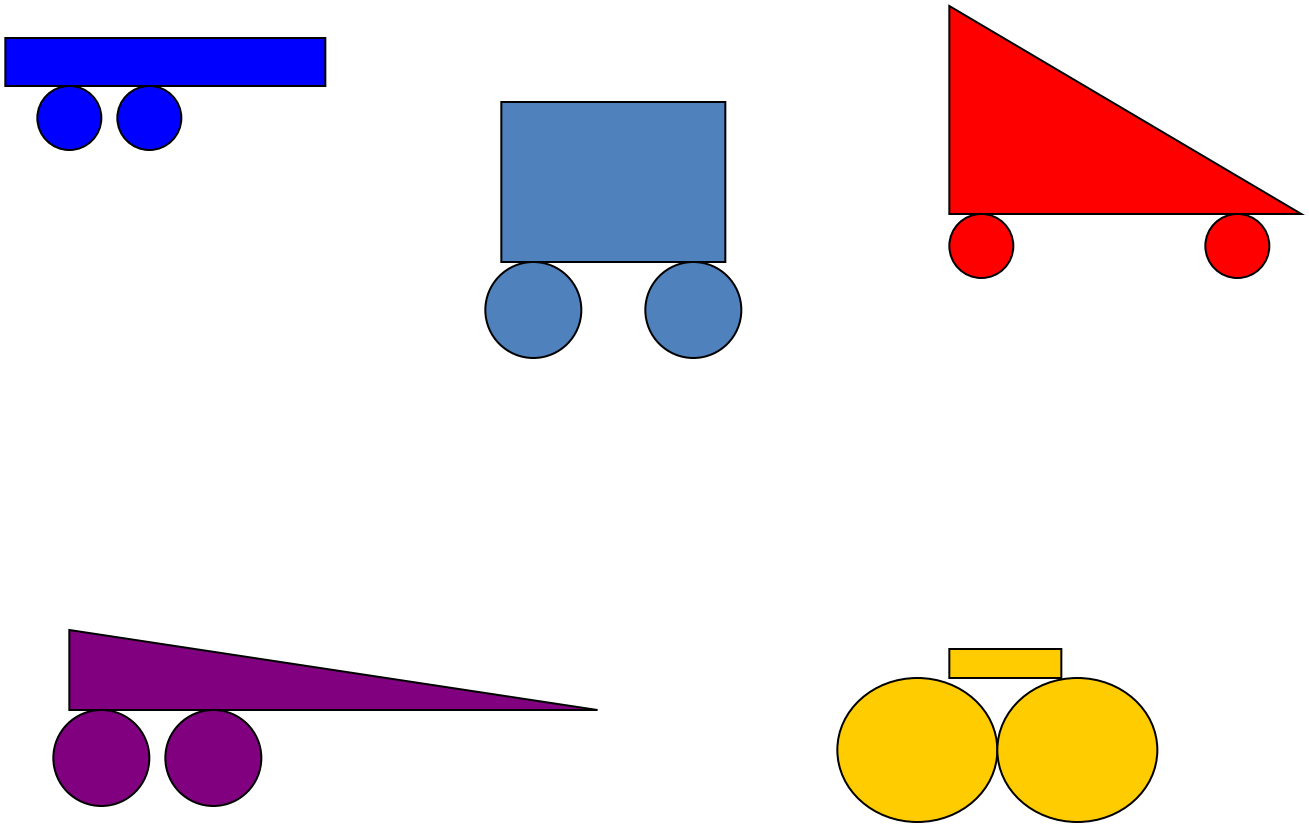
Tuning a GA

- "Typical" tuning parameters for a small problem

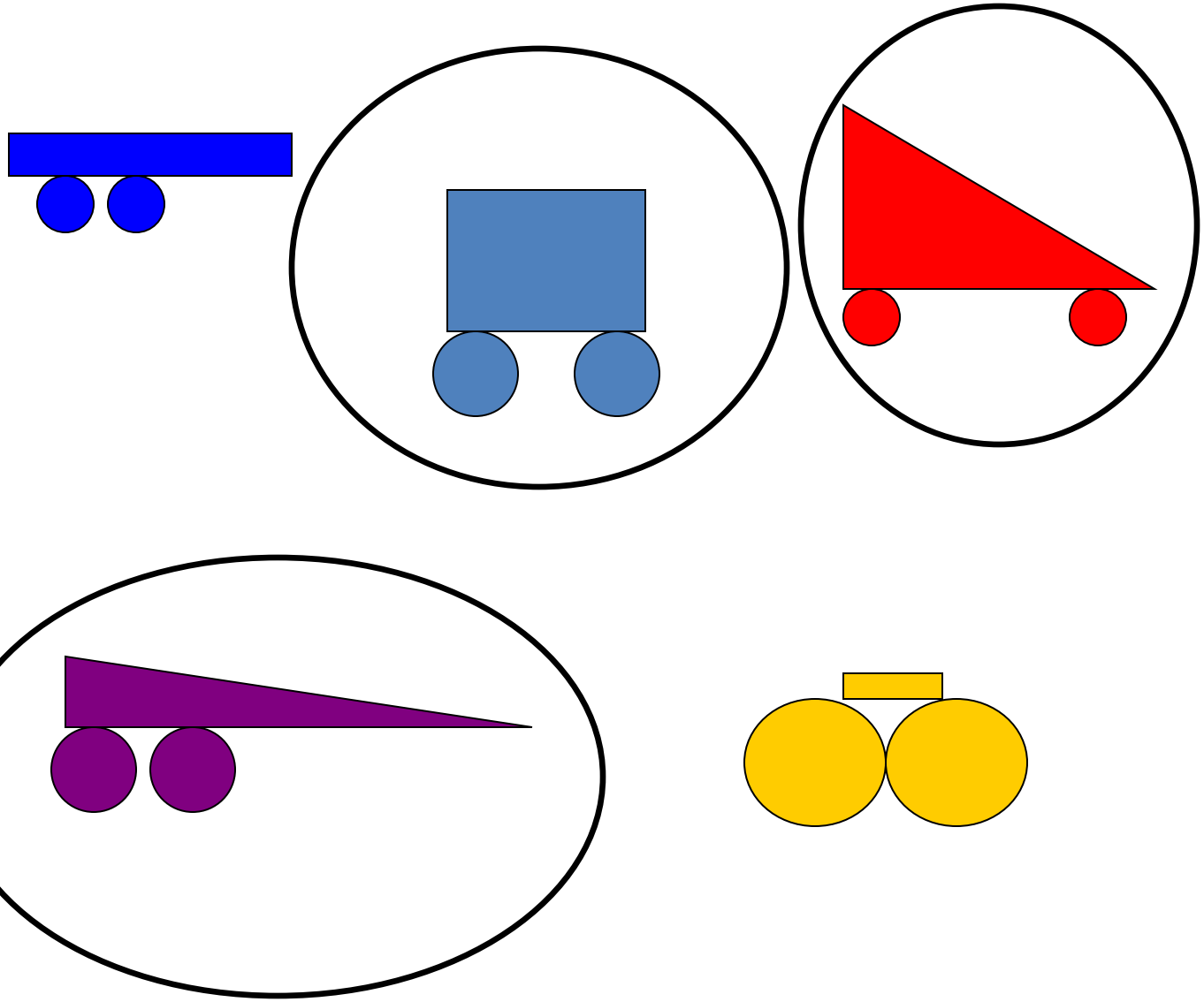
Population size:	50 – 100
Children per generation:	= population size
Crossovers:	0 – 3
Mutations:	< 5%
Generations:	20 – 20,000

- Other concerns
  - population diversity
  - ranking policies
  - removal policies
  - role of random bias

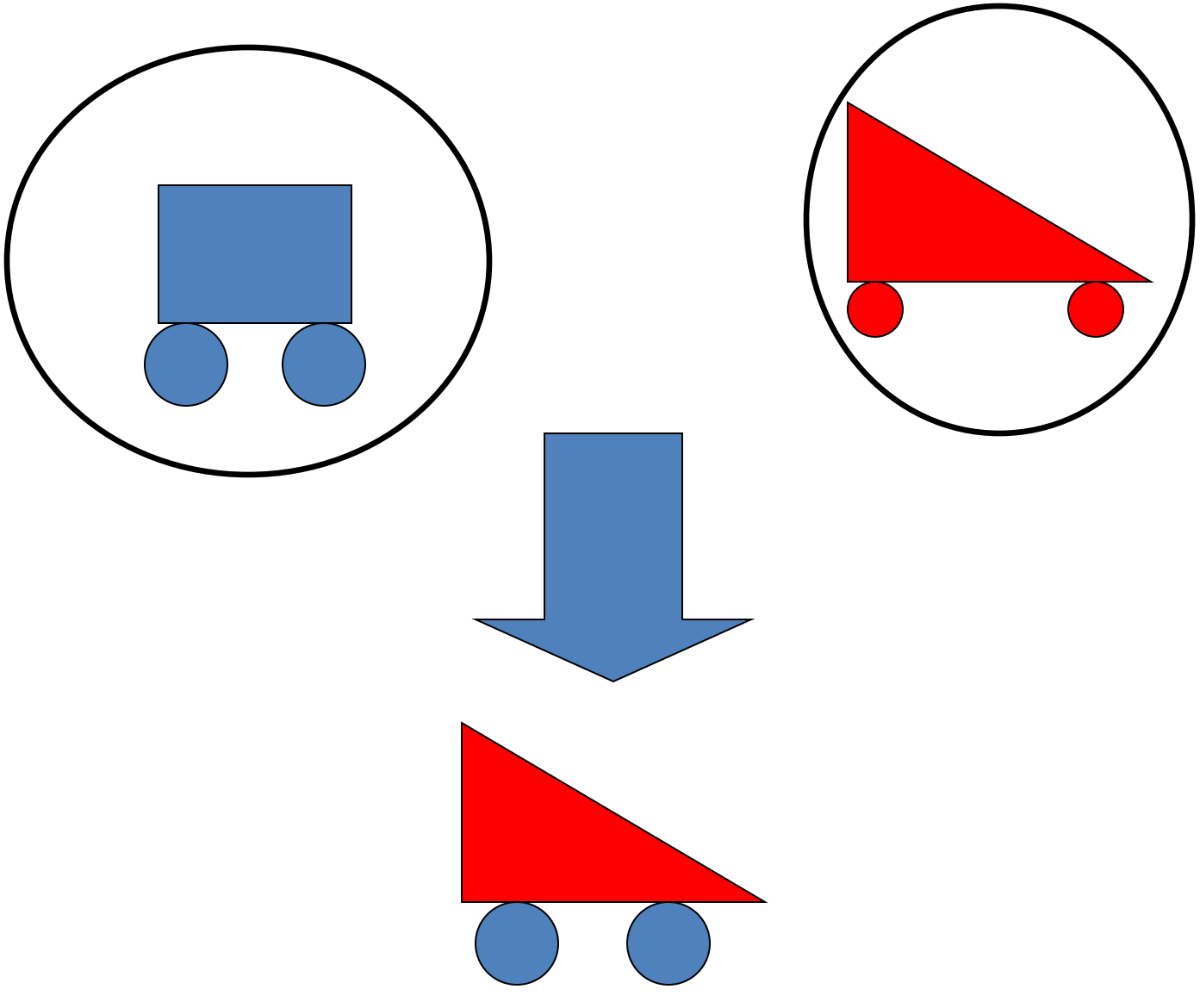
Initial population



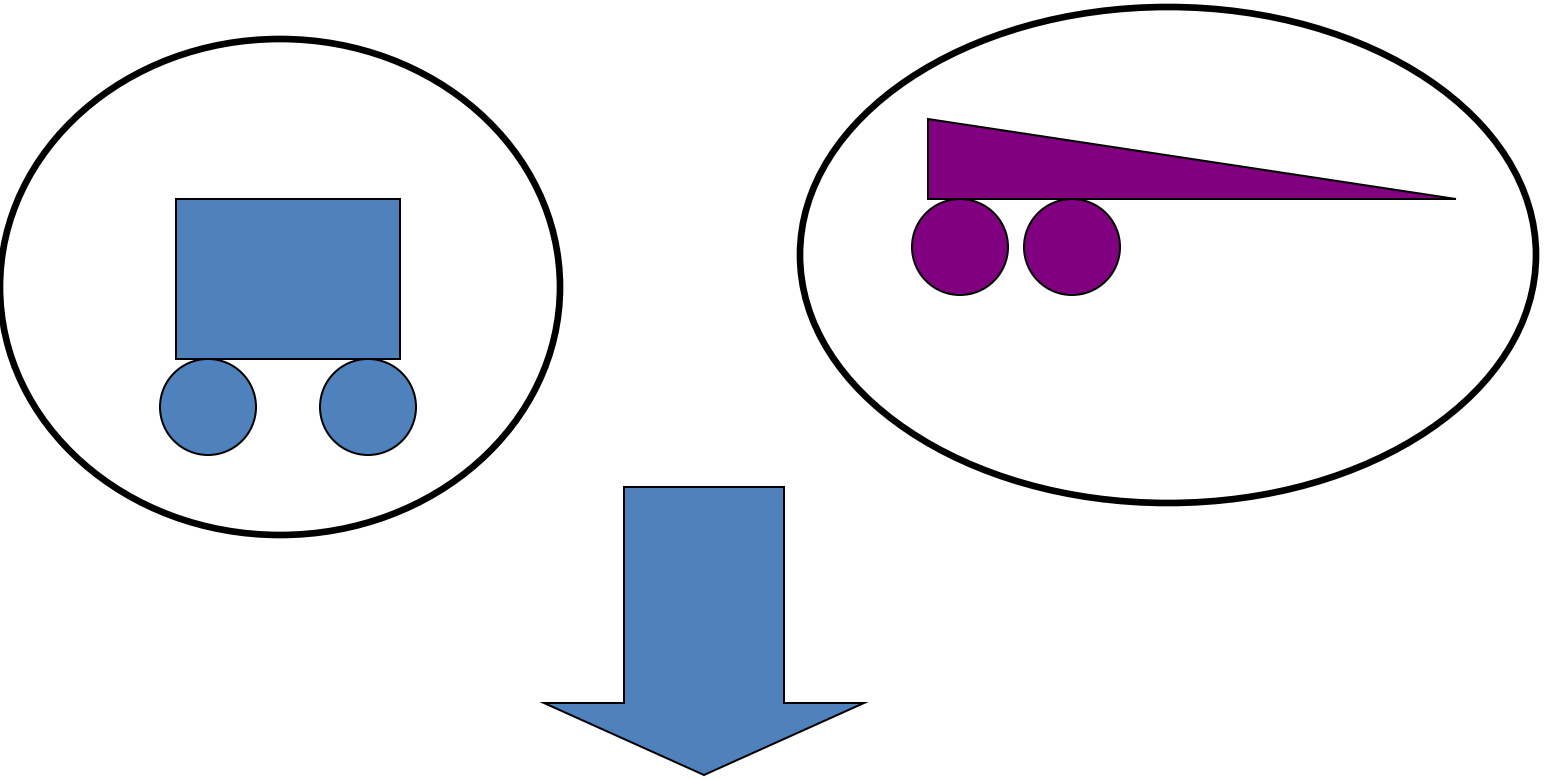
Select



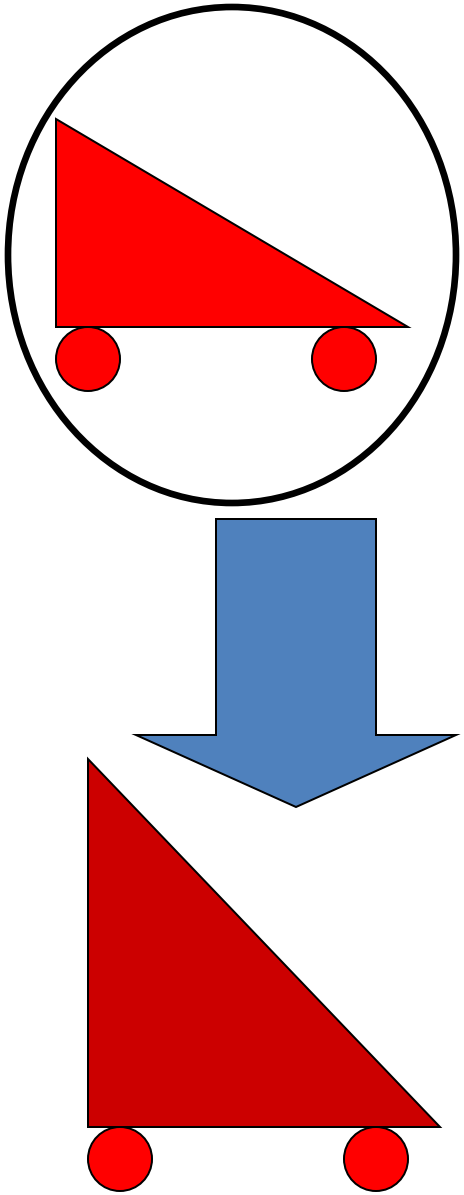
Crossover



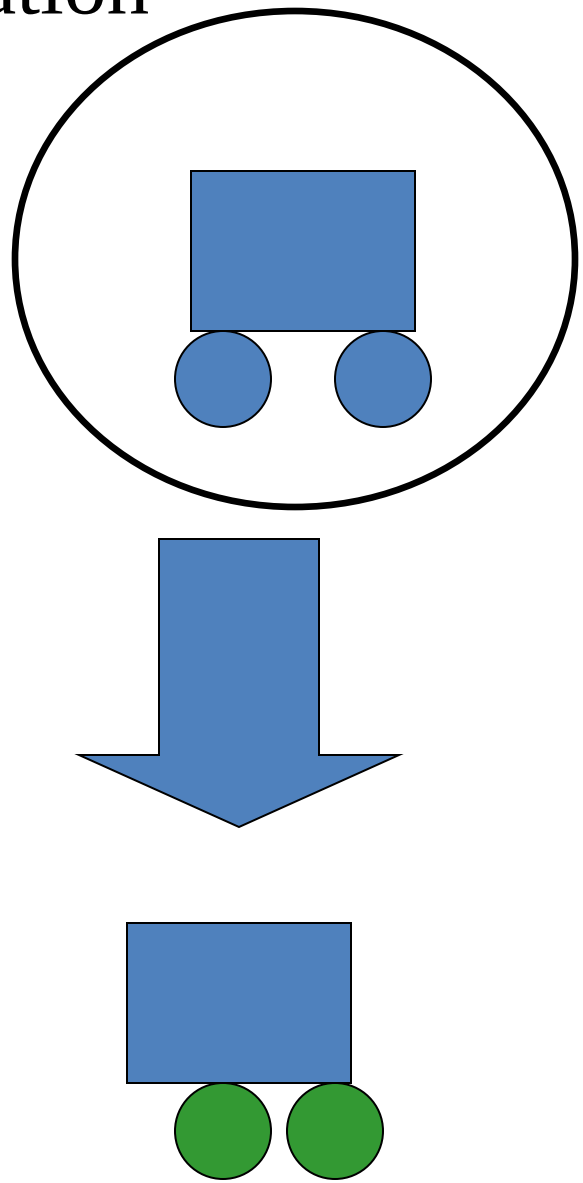
Another Crossover



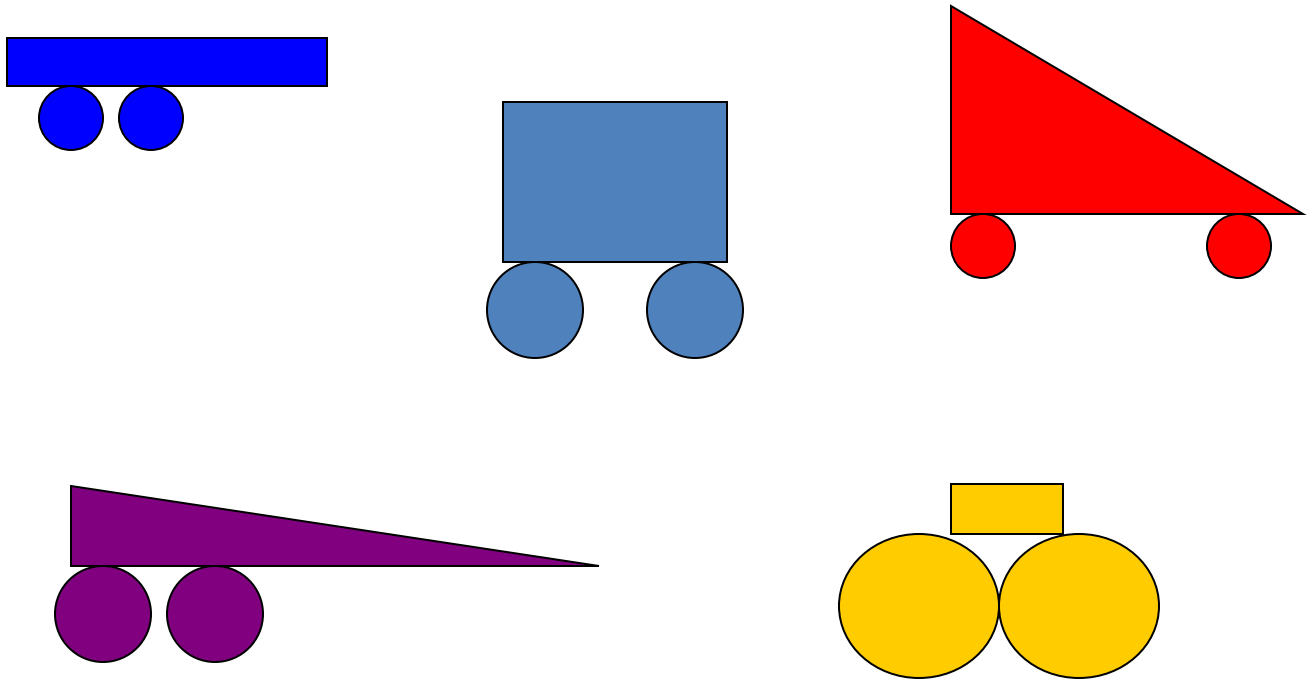
A mutation



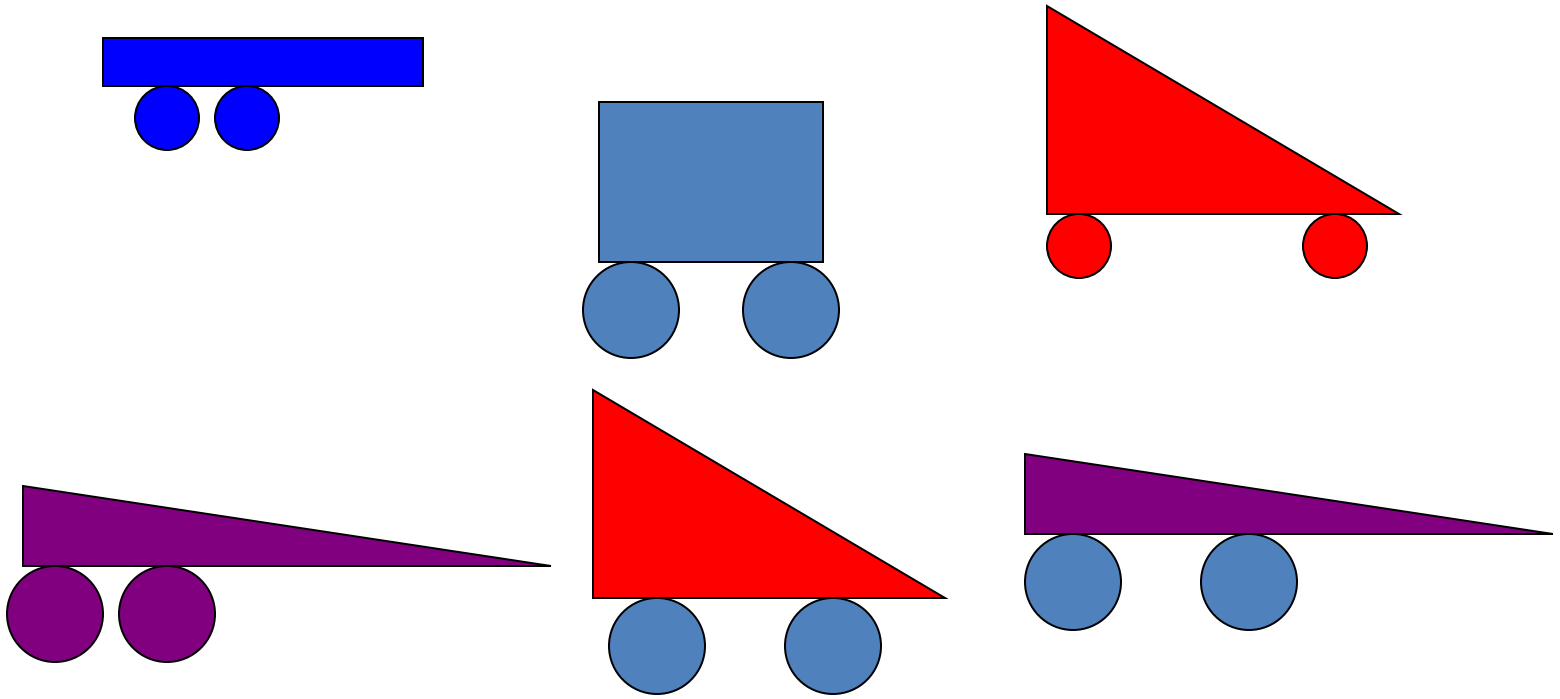
Another Mutation



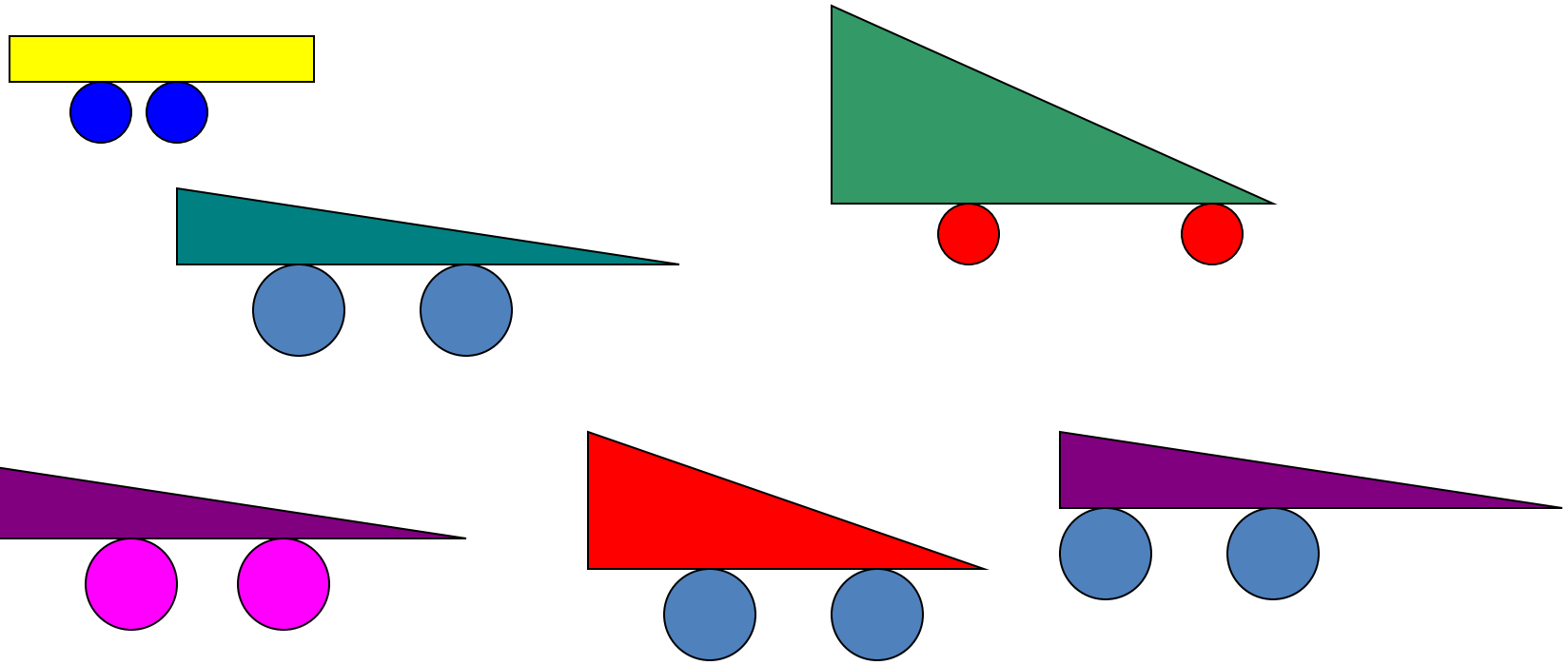
Old population + children



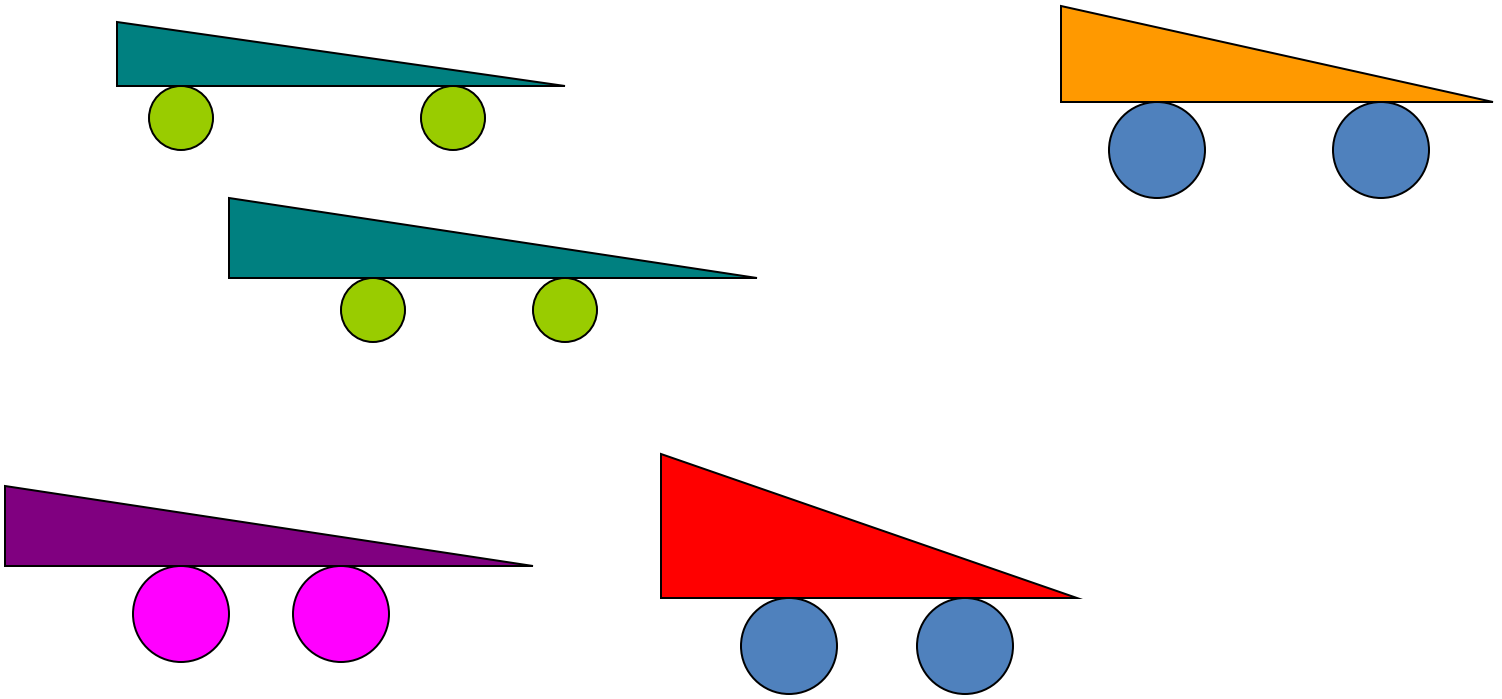
New Population: Generation 2



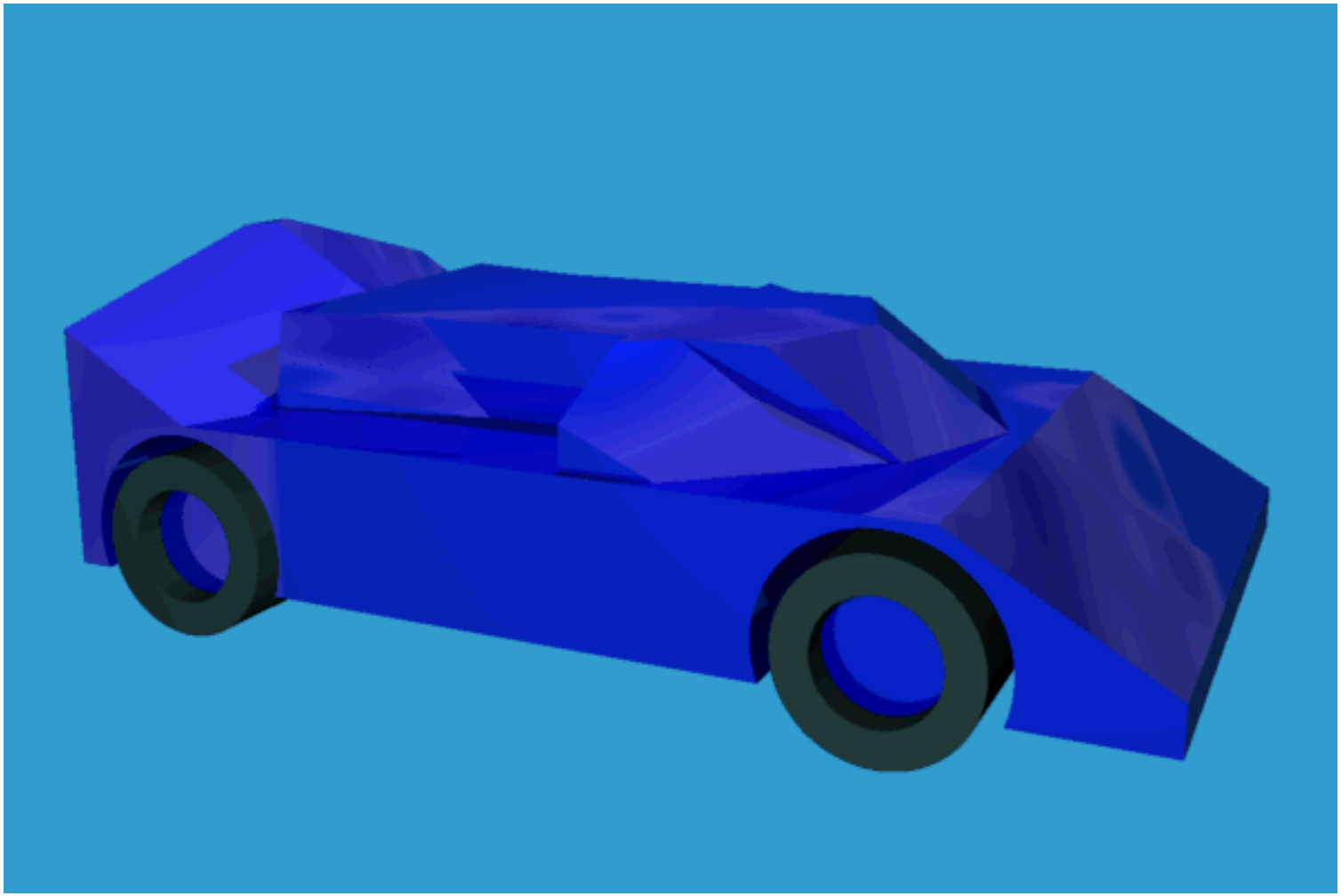
Generation 3



Generation 4, etc …



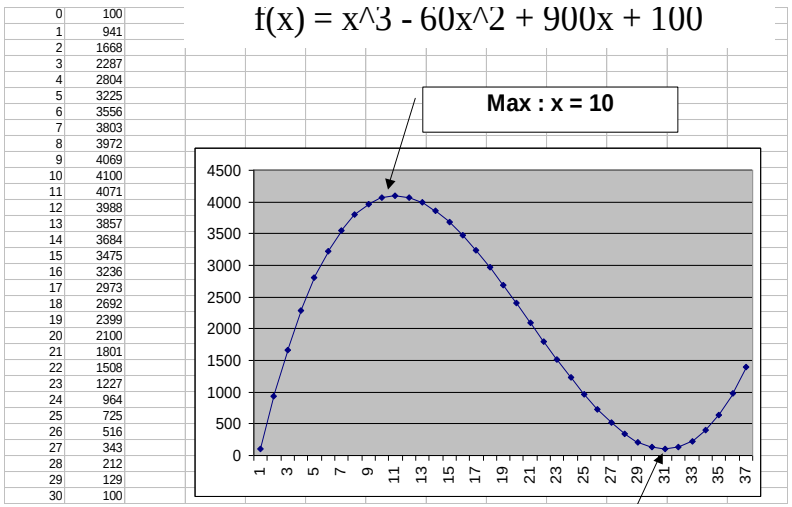
Bentley.s thesis work



Fixed wheel positions, constrained bounding area,
Chromosome is a series of slices
\fitnesses evaluated via a simple airflow simulation

GA Example

- **Crossover probability, $P_C = 1.0$**
- **Mutation probability, $P_M = 0.0$**
- **Maximise $f(x) = x^3 - 60 * x^2 + 900 * x + 100$**
- **$0 <= x >= 31$**
- **x can be represented using five binary digits**



Source: G5BAIM Artificial Intelligence Methods by Graham Kendall

GA Example

- **Generate random individuals**

Chromosome	Binary String	x	f(x)
P_1	11100	28	212
P_2	01111	15	3475
P_3	10111	23	1227
P_4	00100	4	2804
	TOTAL		**7718**
	AVERAGE		**1929.50**

Source: G5BAIM Artificial Intelligence Methods by Graham Kendall

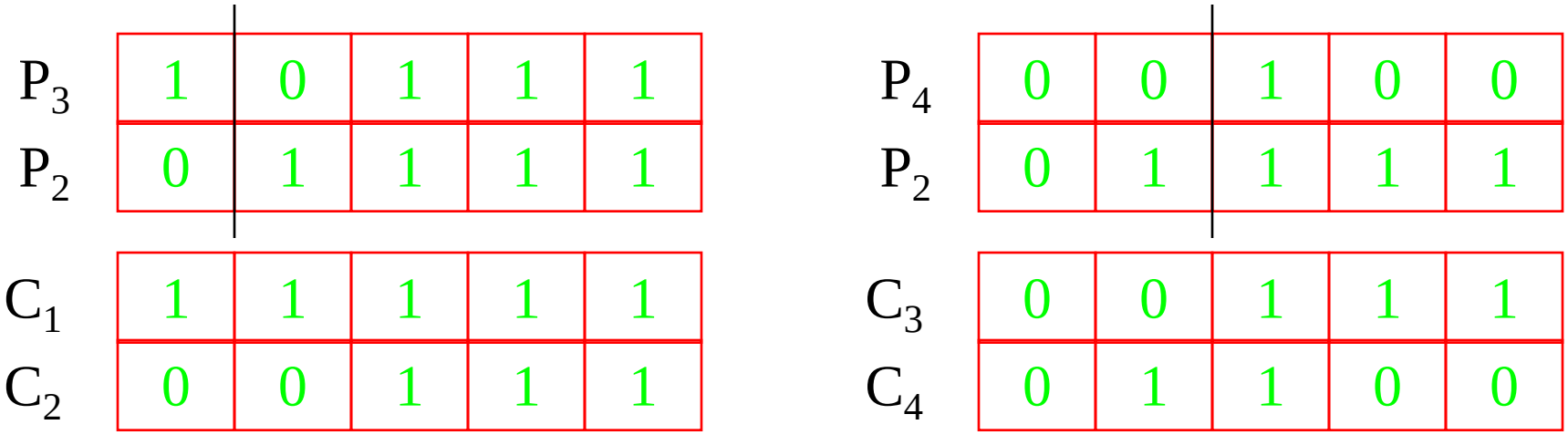
GA Example

- **Choose Parents, using roulette wheel selection**
- **Crossover point is chosen randomly**

Roulette Wheel	Parent Chosen	Crossover Point
4116	P_3	N/A
1915	P_2	1

Source: G5BAIM Artificial Intelligence Methods by Graham Kendall

GA Example - Crossover



Source: G5BAIM Artificial Intelligence Methods by Graham Kendall

GA Example - After First Round of Breeding

- **The average evaluation has risen**
- **P_2, was the strongest individual in the initial population. It was chosen both times but we have lost it from the current population**
- **We have a value of x=7 in the population which is the closest value to 10 we have found**

Chromosome	Binary String	x	f(x)
P_1	11111	31	131
P_2	00111	7	3803
P_3	00111	7	3803
P_4	01100	12	3998
	TOTAL		**11735**
	AVERAGE		**2933.75**

Source: G5BAIM Artificial Intelligence Methods by Graham Kendall

Travelling Salesman Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation

Representation is an ordered list of city numbers known as an *order-based* GA.

1) London	3) Iowa City	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Crossover

Crossover combines inversion and recombination:

Parent1	(3	5	7	2	1	6	4	8)
Parent2	(2	5	7	6	8	1	3	4)
Child	(5	8	7	2	1	6	3	4)

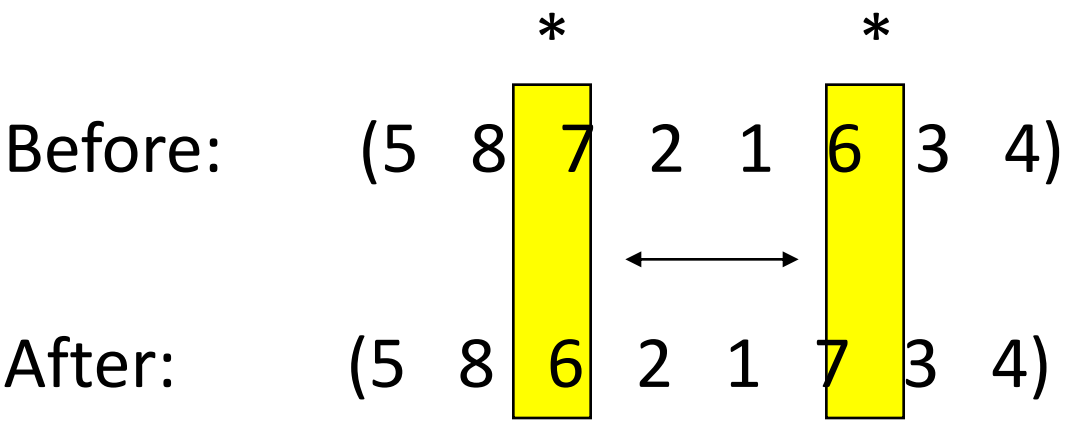
(1) Copy a randomly selected portion of Parent1 to Child

(2) Fill the blanks in Child with those numbers in Parent2 from left to right, as long as there are no duplication in Child.

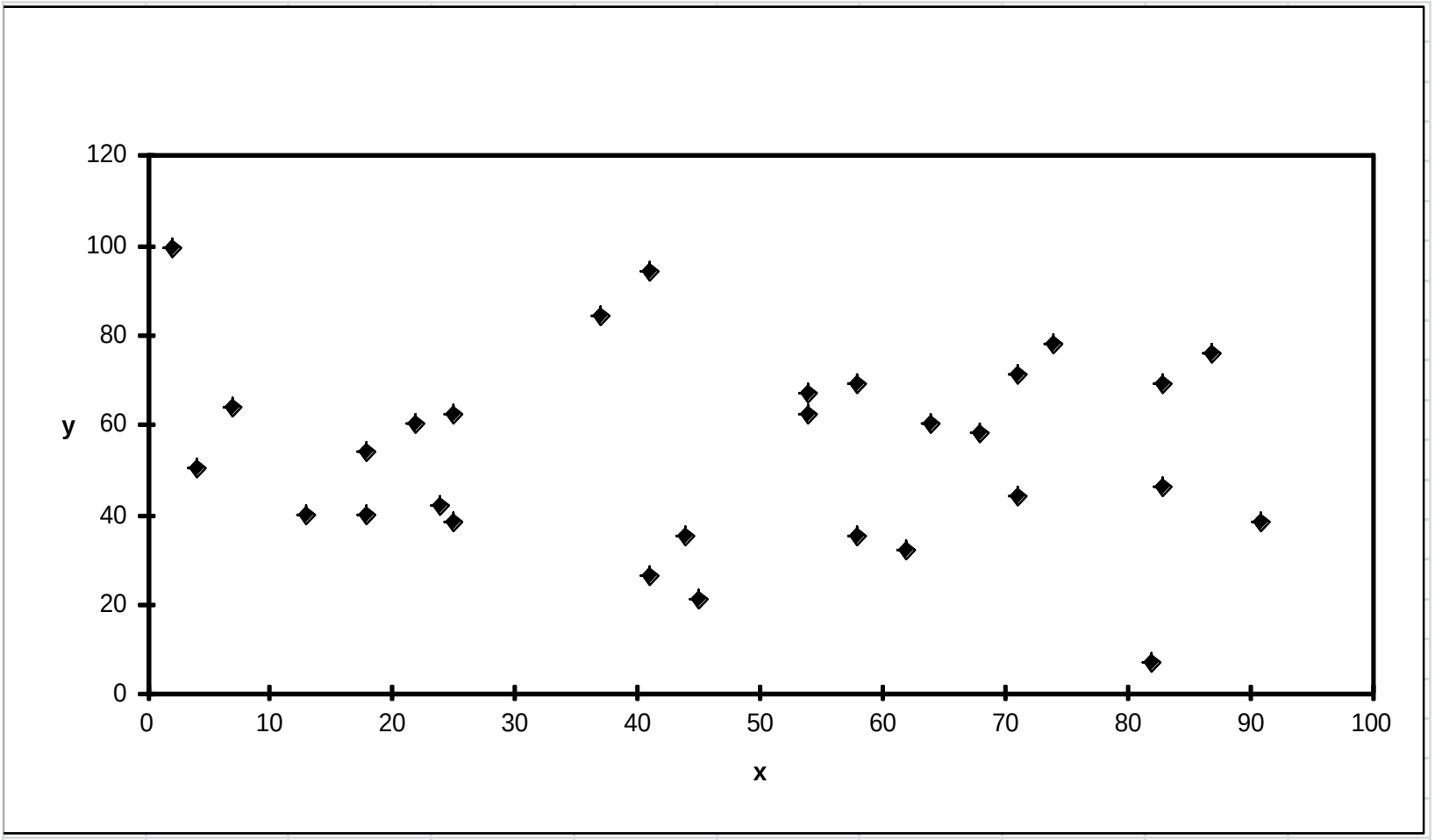
This operator is called the *Order1* crossover.

Mutation

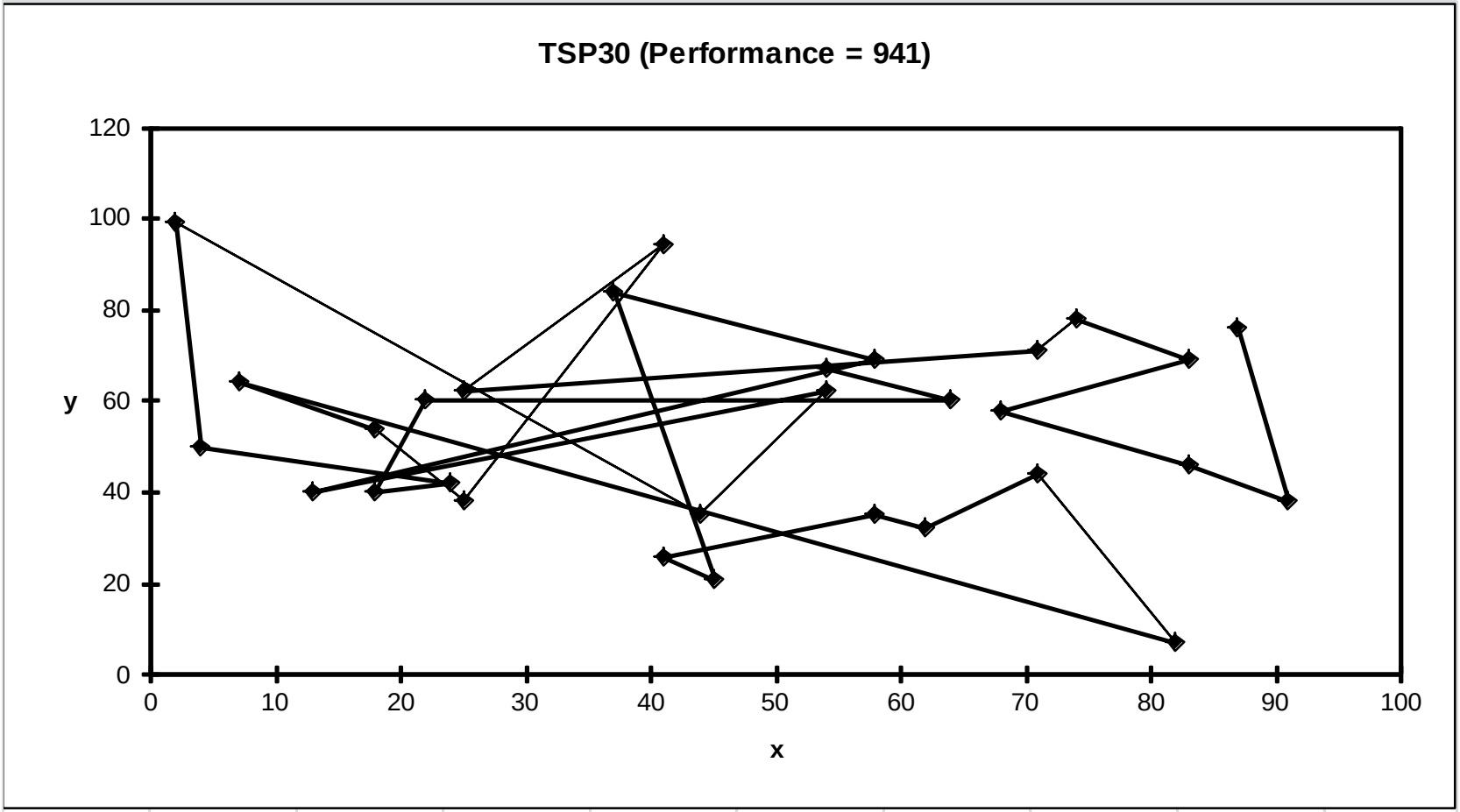
Mutation involves swapping two numbers of the list:



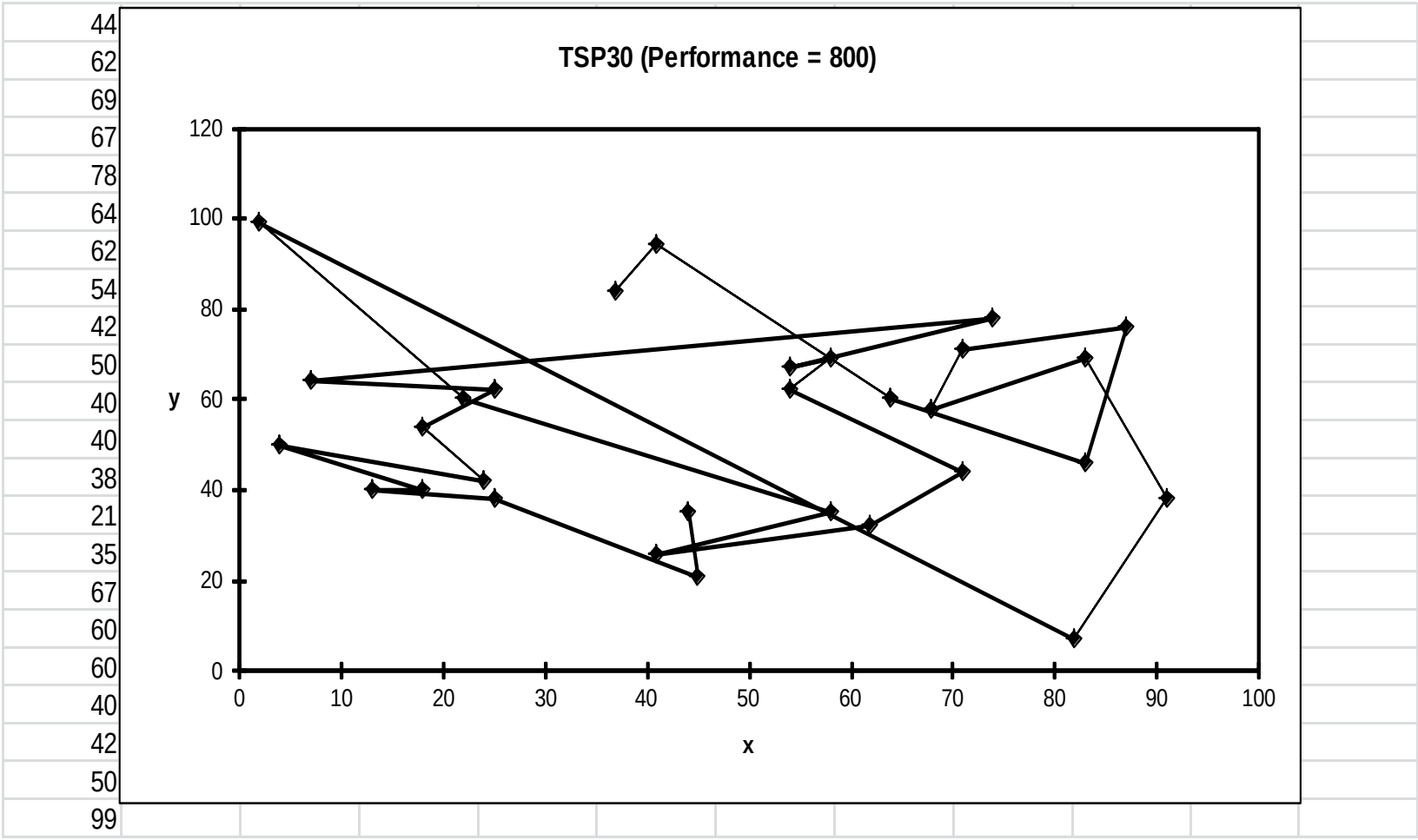
TSP Example: 30 Cities



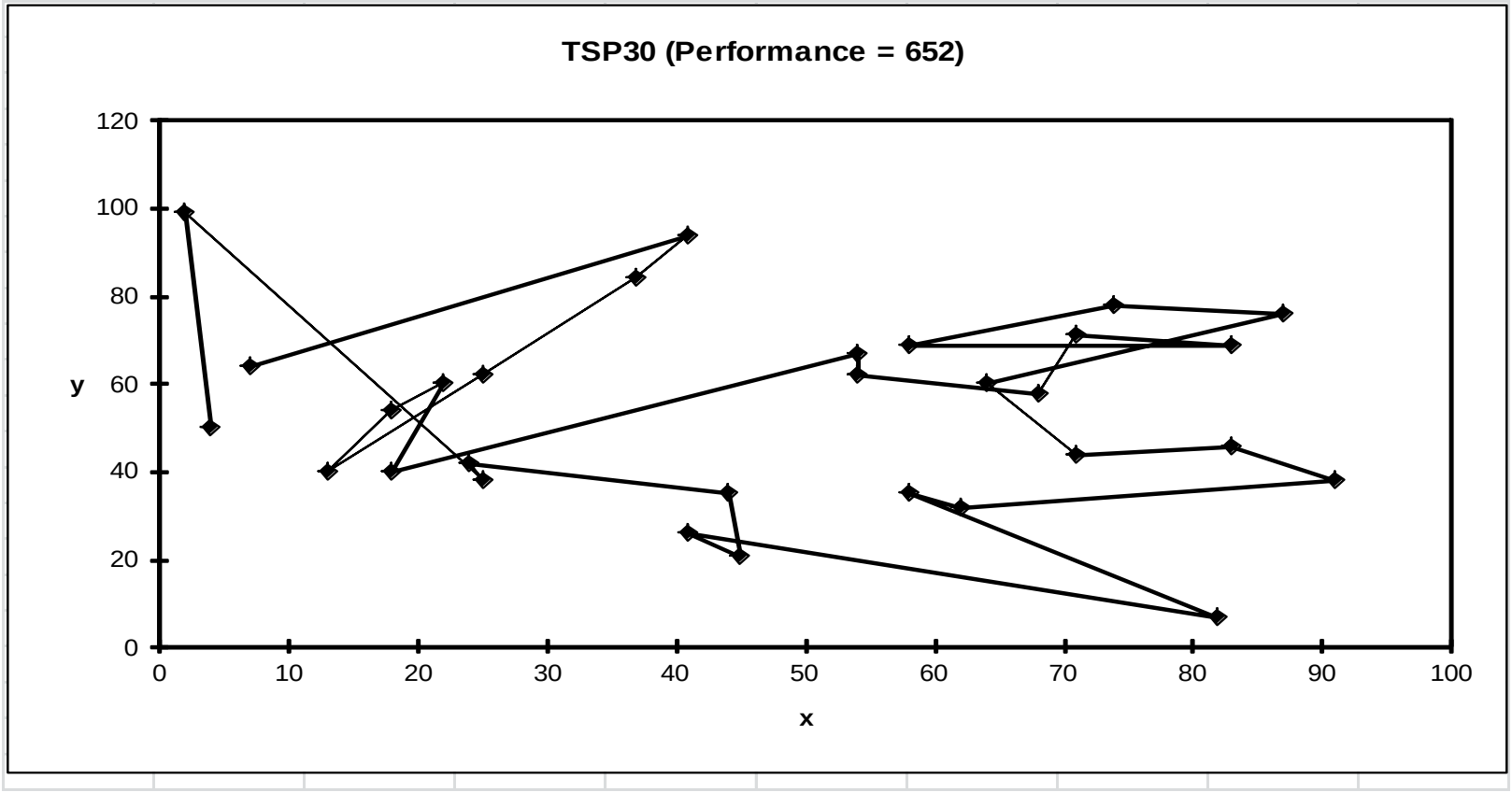
Solution ¡ (Distance = 941)



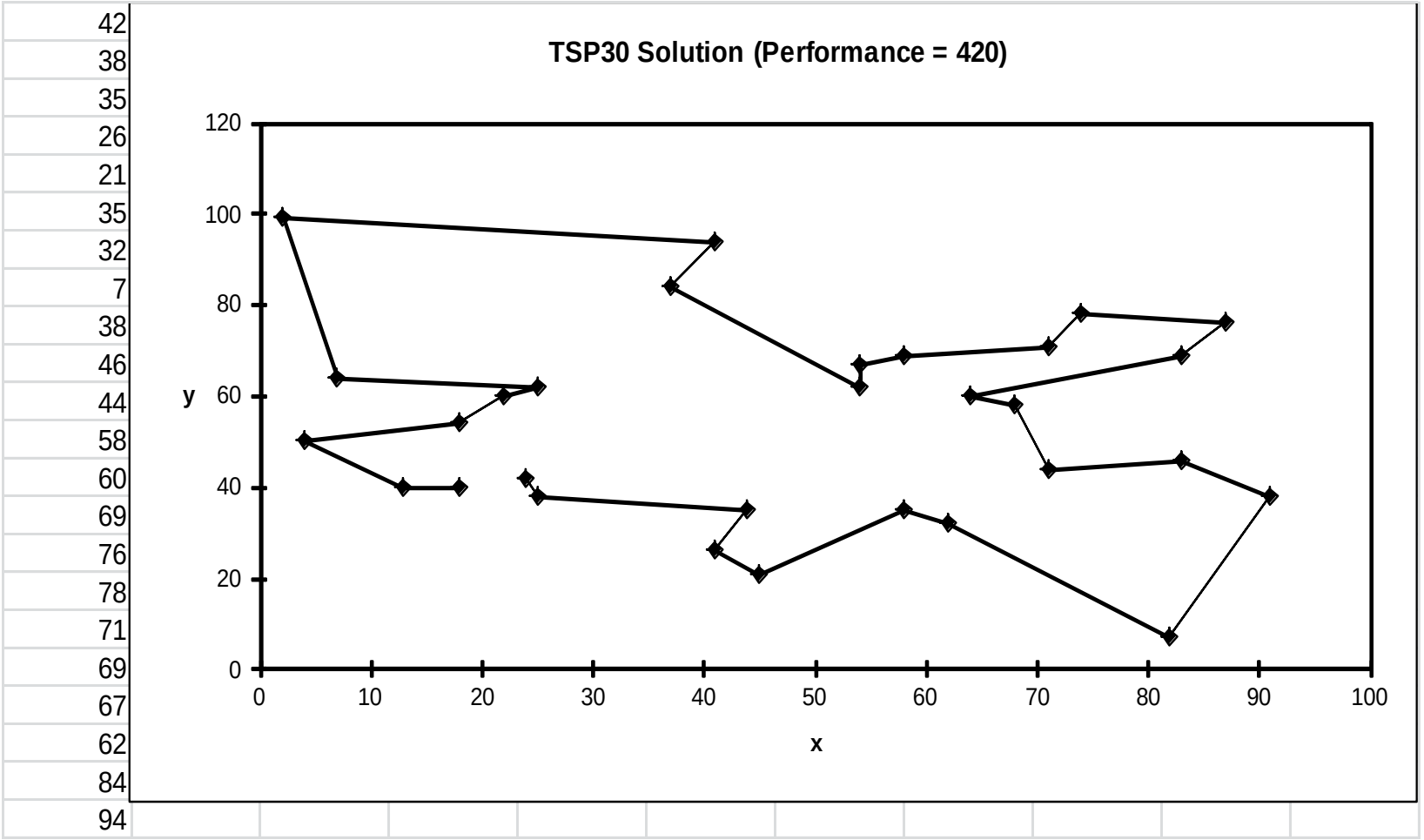
Solution $_j$(Distance = 800)



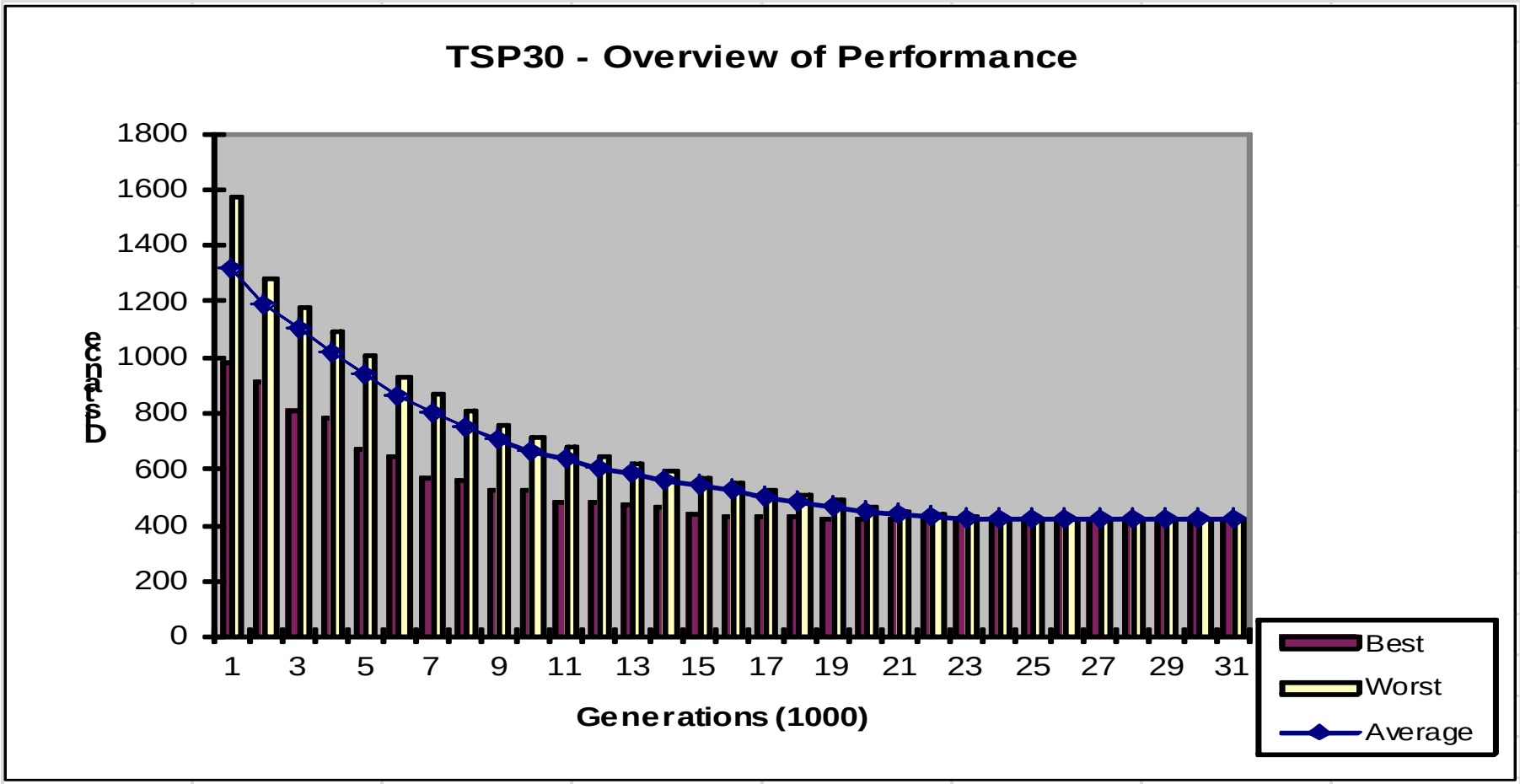
Solution $_k$(Distance = 652)



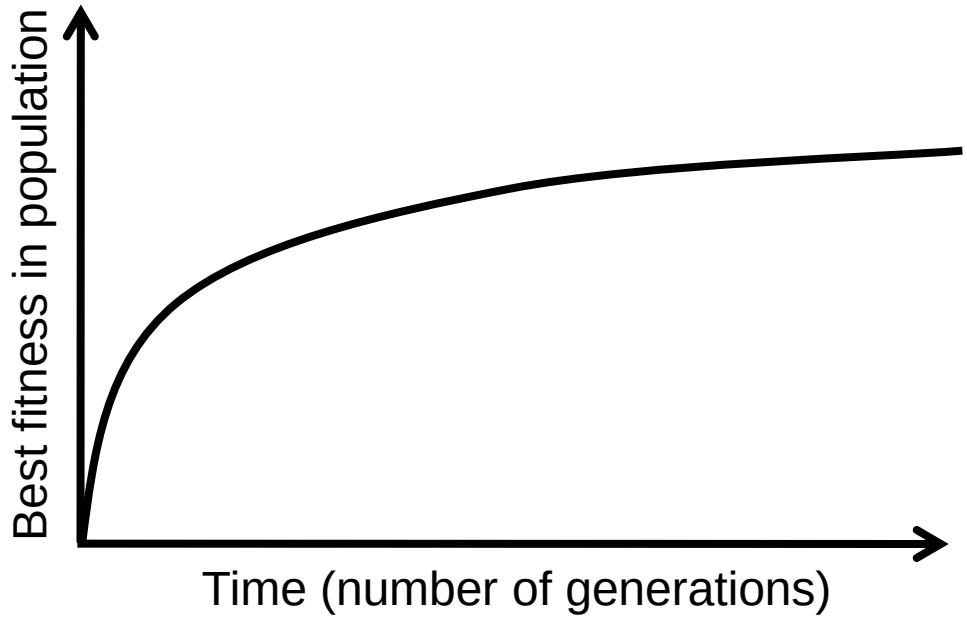
Best Solution (Distance = 420)



Overview of Performance

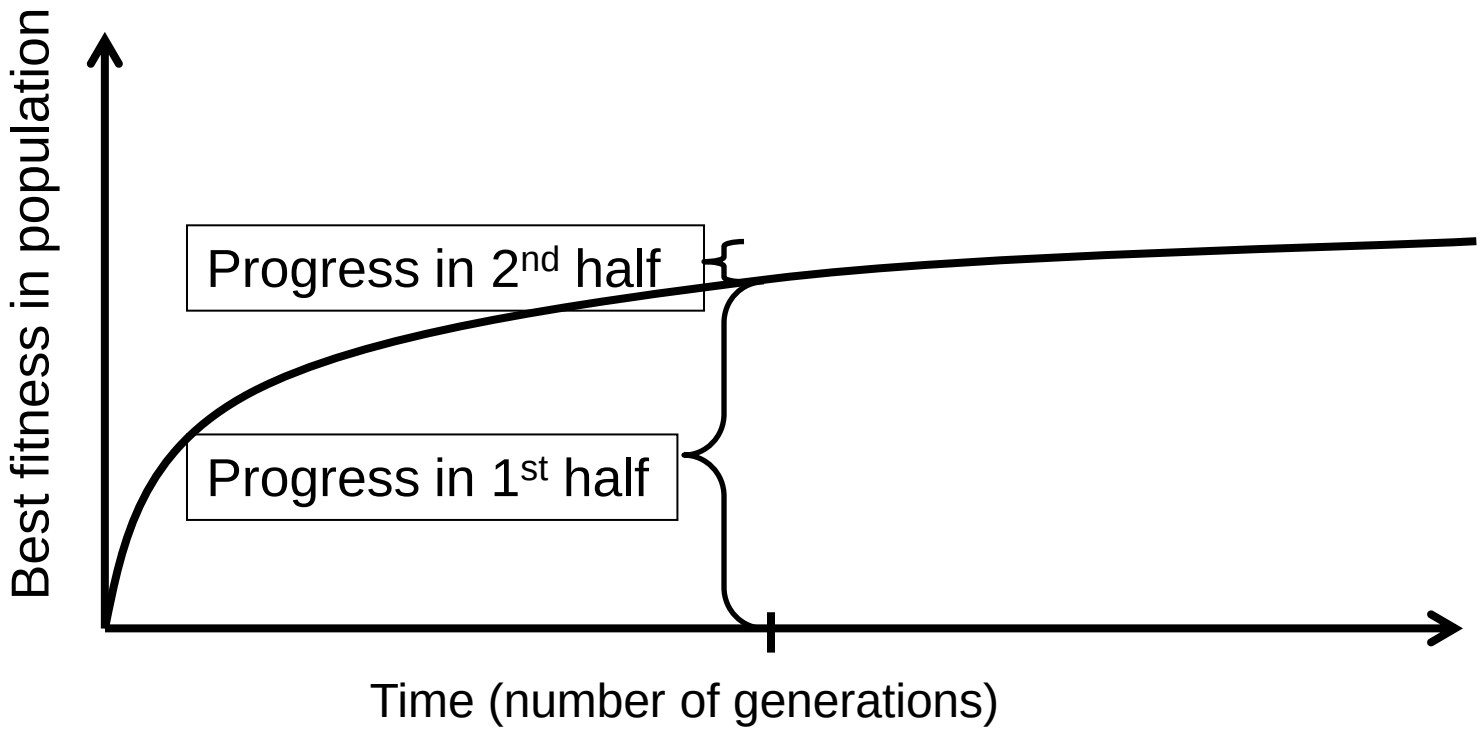


Typical run: progression of fitness



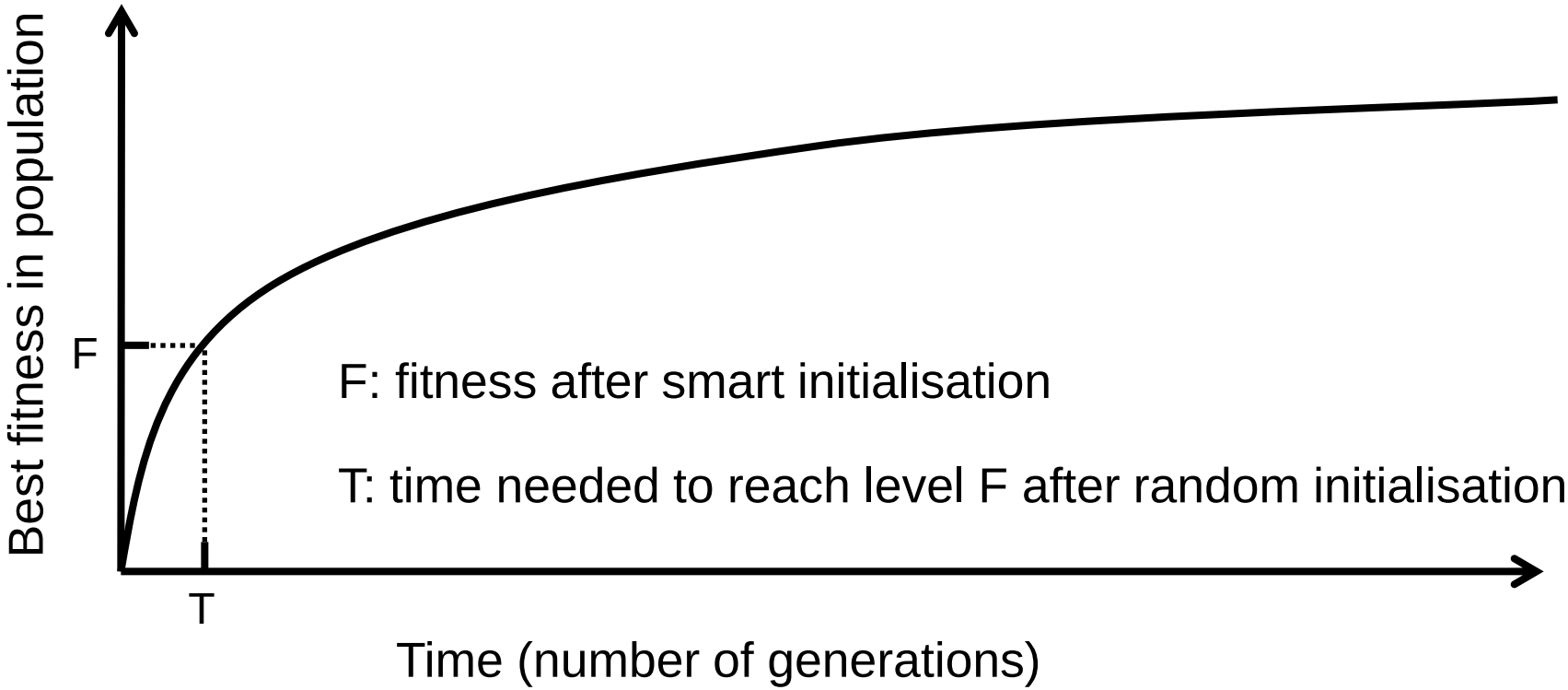
Typical run of an EA shows so-called “anytime behavior”

Are long runs beneficial?



- Answer:
  - it depends how much you want the last bit of progress
  - it may be better to do more shorter runs

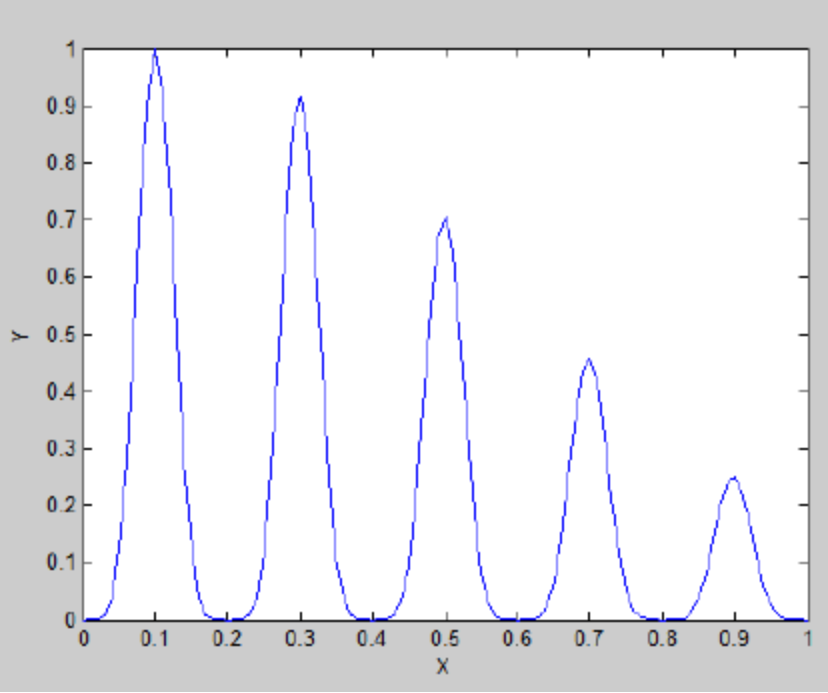
Is it worth expending effort on smart initialisation?



- Answer : it depends:
  - possibly, if good solutions/methods exist.
  - care is needed

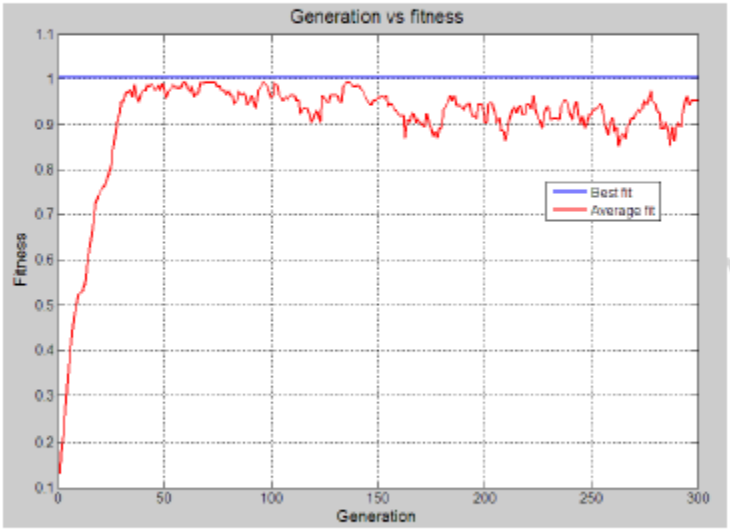
$$Minimize\ f(x) = 2^{-2\left(\frac{(x-0.1)}{0.8}\right)^2} sin^6(5\pi x)$$

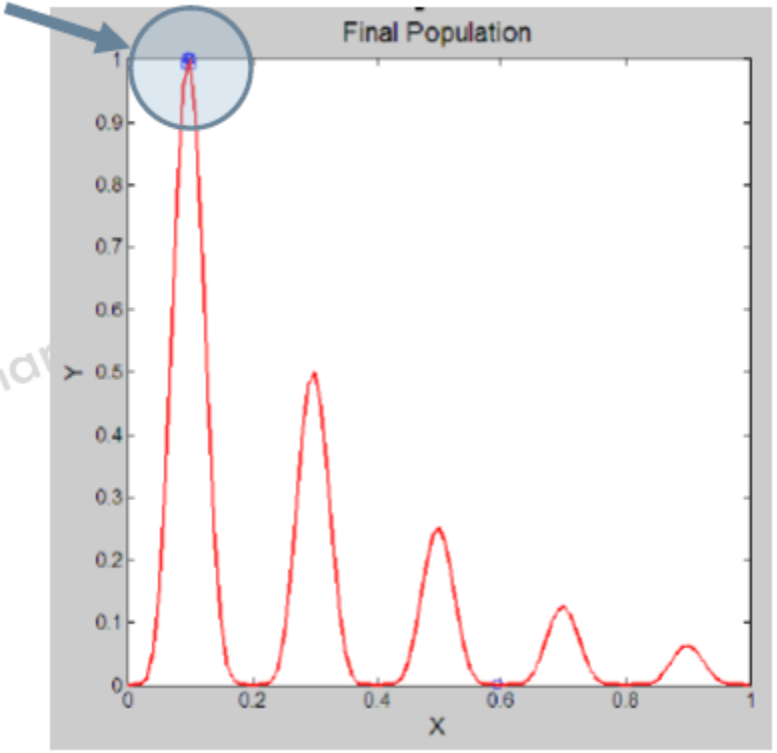
$$0 \leq x \leq 1$$



- After 200 iterations

The population are in and around the global optimal solution





REFERENCES

- Genetic Algorithms in Search, Optimization and Machine Learning by David E. Goldberg.
- Genetic Algorithms + Data Structures = Evolutionary Programs by Zbigniew Michalewicz.
- Practical Genetic Algorithms by Randy L. Haupt and Sue Ellen Haupt.
- Multi Objective Optimization using Evolutionary Algorithms by Kalyanmoy Deb.
- Introduction to evolutionary algorithms by Yu, X. and Gen M., Springer
- G5BAIM Artificial Intelligence Methods by Graham Kendall (PPT Presentation)
- Biological Inspired Computing: EAIntro (PPT Presentation)
- Genetic Algorithms and Evolution Strategies by Julie Leung, Keith Kern and Jeremy Dawson (PPT Presentation)
- https://www.tutorialspoint.com/genetic_algorithms/

Thanks