University of BRISTOL

Introduction to AI

Week 23

Dr. Ayush Joshi

Last week's (before Easter) content

Week 22:

- Agent based systems
- MDP
- Reinforcement learning

Nature inspired computation

Week 23:

- Natural computation
- Evolutionary algorithms

Nature inspired computation

- Algorithms inspired by nature!

1. Taking ideas from nature to develop 'novel' problem solving techniques
2. Recreate/emulating natural phenomenon in computers.
3. Use 'novel natural materials' to perform computation.

- These 3 'related' approaches constitute 'natural computation'

- Let’s briefly look at these.

Nature inspired computation

- Algorithms inspired by nature

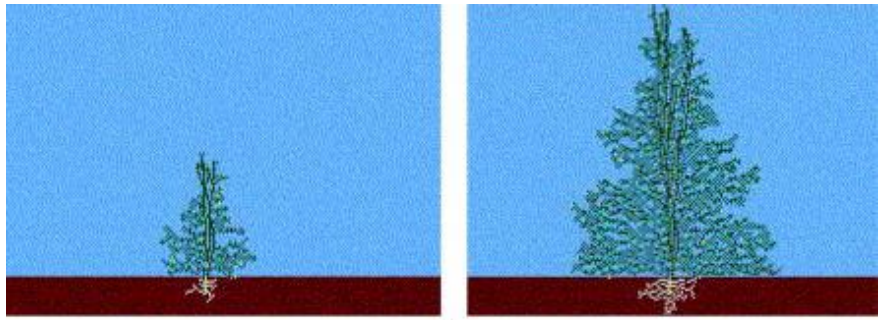
1. Taking ideas from nature to develop 'novel' problem solving techniques
   - Oldest and most popular technique (among natural computing).
2. Two main objectives:
   - Devise theoretical models which can reproduce the functioning of the natural phenomenon studied.
   - Study of the natural phenomenon to develop algorithms which can solve complex problems.
3. Well known examples are:
   - Neural networks (You have studied them in this unit before!)
   - **Evolutionary algorithms. (We will study them in detail here)**
   - Swarm intelligence.
   - Artificial immune systems

Synthesis of nature in computers

- Synthetic approach to simulate natural phenomenon in computers

1. Two main approaches:
   - 'Artificial life' techniques
     - Man made systems that have characteristics of natural systems.
   - Tools to study 'Fractal geometry of nature'
     - Cellular Automata (developed in late 1940s) - mathematical systems which are spatially and temporally discrete, characterised by local interactions and inherent parallel form of evolution in time.
     - L systems - a formal system to simulate development of multicellular organisms.

2. Usage
   - CA: insights into chemical reactions, crystal growth, seashells, phenomenon in vehicular traffic.
   - L-systems: reconstruction of extinct plants, structural model of trees in ecosystem simulations, crop yield prediction, musical composition.
   - ALife – applications in art, games, evolution of language, robotics.



(a)

(b)

a) Tree growth generated by CA.

CAFUN (www.cafun.de)

b) 3D rendering of a plant simulated using L-system

Image from: https://www.sciencedirect.com/science/article/pii/S1571064506000315

Computing with new materials

- New computing methods based on natural materials (other than silicon!)
- Molecular computing
- Quantum computing

- We do not cover these here.
- Big enough topics to warrant modules on their own!
- Just a flavour of what is out there so you can explore.

Problems of interest!

- Setting the stage!
- Black box model of computation system -
  - System waits for some input.
  - Upon input, system processes that using some model whose details not known (hence black box!)
  - This model represents aspects of the world relevant to the application.
  - e.g. Model could be formula that calculates route length given some locations, statistical tool that estimates likelihood of rain given some data, etc.
  - After processing, system returns an output.
- Based on this, we can have 3 problem types
- Optimisation : model and output known, task is to find input leading to output.
- Modelling : inputs and outputs are known and a system/model is sought which links correct inputs to output.
- Simulation: We know some inputs and model and want to the outputs.

Problems of interest!

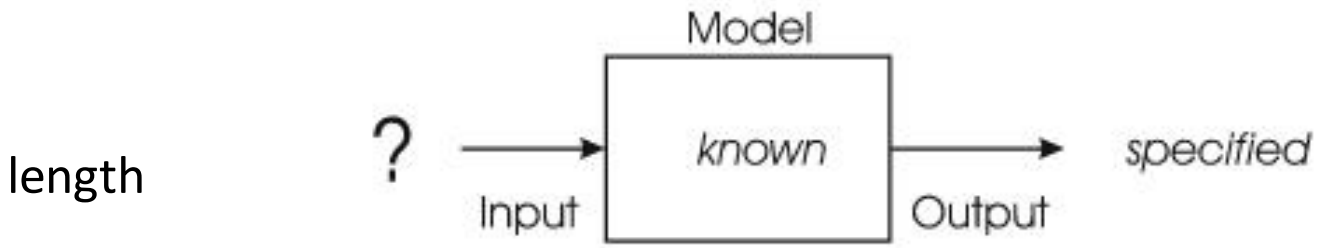
- Optimisation : model and output known, task is to find input leading to output.

Example: Travelling salesman problem

Input = Sequence of cities

Model = A formula that computes length

Output = length of the tour.

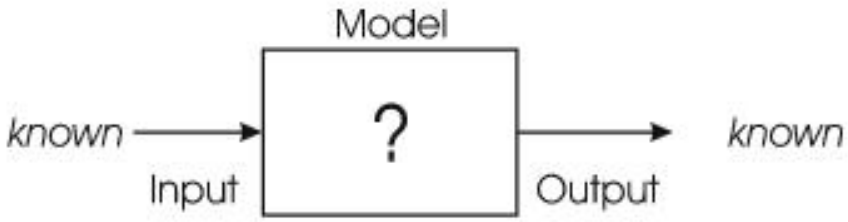


Problem = find the input with a desired output! Sequence with optimal (minimum) length

Problems of interest!

- Modelling : inputs and outputs are known and a system/model is sought which links correct inputs to output.
- Stock exchange example

- Input = economic indices (employment rate, gold price, etc)
- Output = Dow-Jones index price
- Model = ?



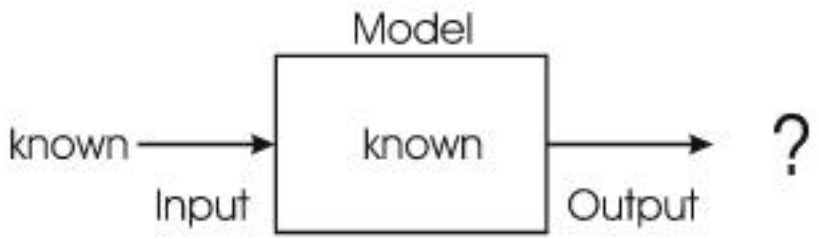
- We want to find the formula that links the input to the output!

Problems of interest!

- Simulation: We know some inputs and model and want to the outputs.
- Electronic circuits example

- Model = system of equations that describe functioning of a circuit
- For any given signal, this model can compute output!

- Simulation is cheaper than actual construction of the model.



Evolutionary algorithms (Background)

- As the name suggests – inspired by the process of 'evolution in nature'.
- Not surprising that 'evolution' was selected as a process -
- Power of evolution evident in the diverse species that make up the world, tailored to survive in their niche.
- The fundamental metaphor of evolutionary computation relates 'natural evolution' to a particular style of problem solving - 'trial and error'
- Broadly -

1. A given environment is filled with a population of individuals who must strive for survival and reproduction.
2. Fitness of an individuals is based on its environment and how well they achieve their goals.
3. Over time the population moves towards 'good/higher fit' individuals.

Evolutionary algorithms (Brief history)

- 1940s – Turing proposed 'genetical/evolutionary search' (before computers!)
- 1962 – Bremermmann executed experiments on optimisation through evolution and recombintation.
- 1960s – **Three** different implementations of the basic idea in different places!
- In USA – **Evoluionary programming** (Fogel et.al.) and **Genetic algorithms** (Holland)
- In Germany – **Evolution strategies** (Schwefel, et.al.)
- For 15 years they developed independently!
- Since early 1990s – they are viewed as representatives of one technology: **evolutionary computing/algorithms.**
- Also 1990s – **Genetic programming** (Koza)
- In modern terms- they all constitute evolutionary computing and the algorithms are called evolutionary algorithms.

Biological inspiration!

- Darwinian Evolution
- Theory explains origin of

Biological diversity.

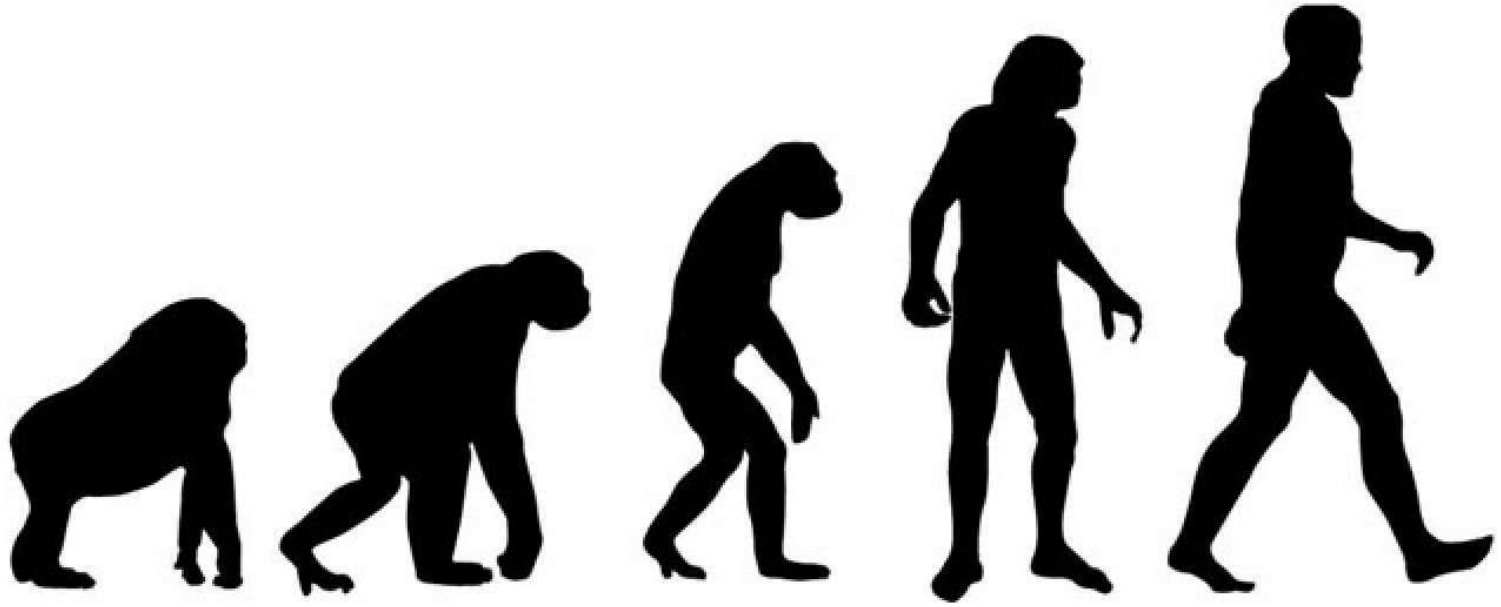
- Natural selection

(survival of the fittest)

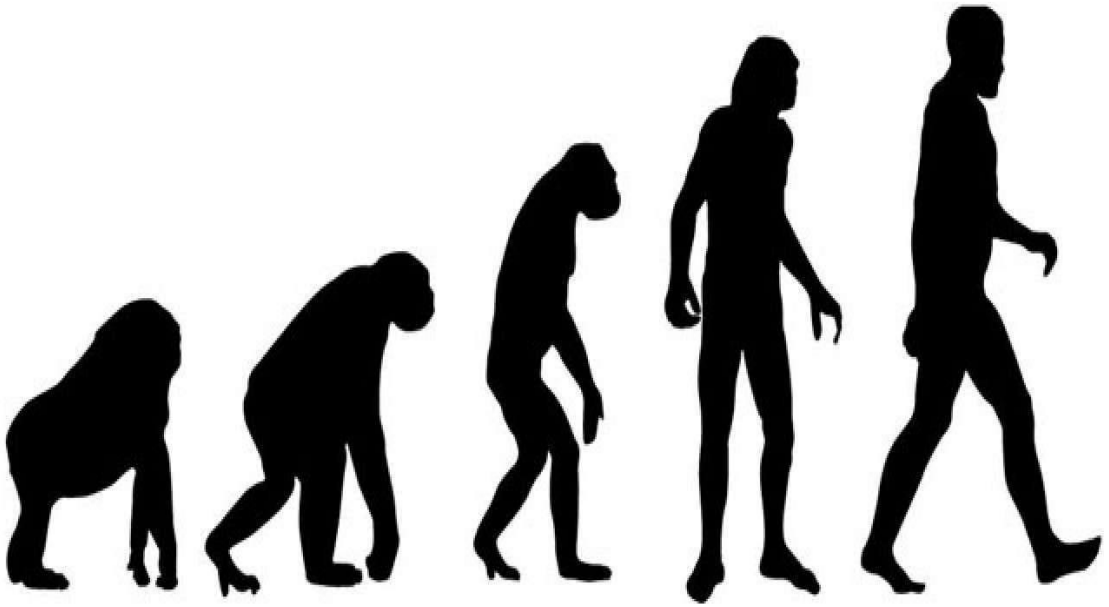
- Mutation

(new traits introduced)

- Eg. Color of eyes!

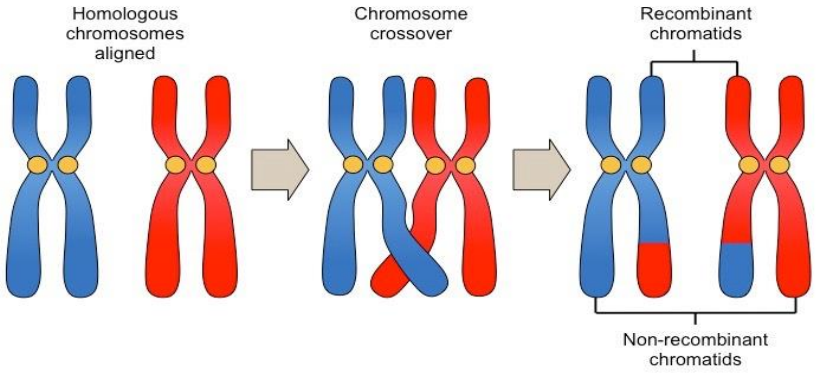


Biological inspiration!



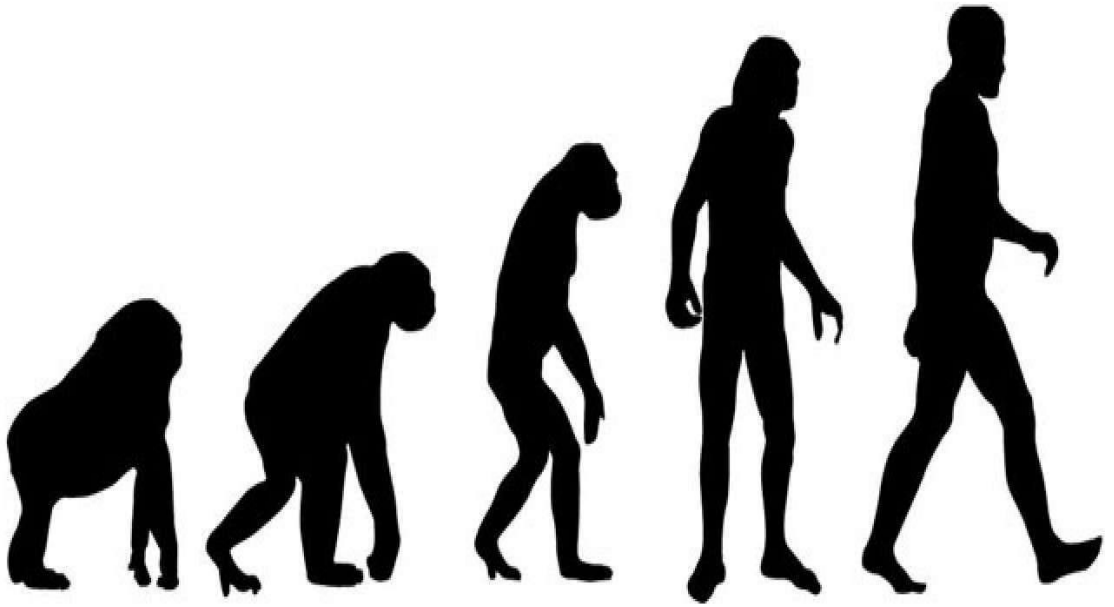
- Darwinian Evolution
- In the environment, resources are limited.
- All individuals compete for resources.
- Most effective ones, fit for the environment.
- Competition based selection is one of 2 main concepts of evolutionary progress.
- Other is 'phenotypic traits'.
- These are 'features' physical or behavioural that affect its response to the environment.
- Each individual is unique in these features.
- Good features = higher chance of creating offspring, bad features = individual dies without offspring.
- **Darwin's insight =** small random variations in phenotypes occur at reproduction!
- In this way, new combinations of traits get introduced and evaluated = progress of evolution!

Biological inspiration!



- Genetics
- Underlying process behind 'phenotypes' and Heredity.
- Phenotypes (outside) are represented at a 'Genotypic' level (inside).
- Genotype encodes phenotype!
- Genes are 'units' of inheritance encoding the physical traits.
- A gene can have many possible values and 'a possible' value is called an '**allele**'.
- Genetic material is contained in 'chromosomes' (46 in Humans, 23 pairs).
- During reproduction, combination of maternal and paternal chromosomes (23 each in humans!) results in new offspring with a full set of chromosomes (23 pairs in humans)
- Combination of features form 2 individuals is called **Crossover** in evolutionary computing.
- Note! - this is a very simplified and the biology is complicated!

Basic Model summary!



- A population consists of a number of individuals.
- They are the units of selection.
- I.e. their reproductive success depends on how well they are adaptive to the environment relative to the rest of the population.
- As more successful individuals reproduce, occasional mutations give rise to new individuals to be tested.
- As time progresses, there is a change in the constitution of the population.
- Therefore, population is the 'unit' of evolution!

Evolutionaty computing – Why?

- Developing algorithms is one of the central themes of Computer science and maths.
- Engineers have always looked at nature's solutions for inspiration!
- When looking at powerful natural problem solvers – 2 candidates
  - Brain
  - Evolution

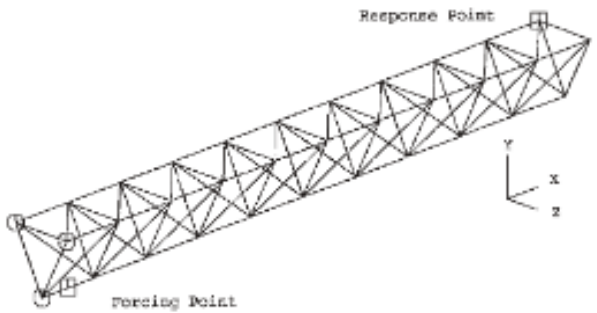
- Another motivation - time available for problem analysis and tailored algorithm design has been decreasing (due to growing demand in automated problem solving)!
- Parallel trend is increase in complexity of problems to be solved.
- These 2 imply = urgent need for robust algorithms with 'satisfactory' performance.
- Human curiosity! 'Playing around with evolution' to see and understand how it works.

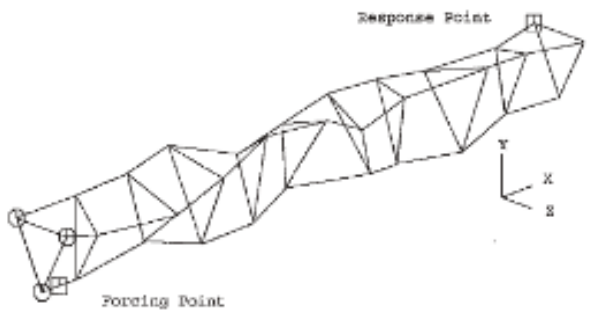
Evolutionaty computing – practical examples!

- University class scheduling. (Optimisation!)
  - 2000-5000 events per week!
  - Must be allocated a day, time and room.
  - Clashes must be reduced! (students in 2 classes at the same time or room used for 2 lectures at the same time)
  - Producing feasible results is HARD!!!
  - In addition to feasible results, we also want optimisation with respect to users.
  - Students/lecturers may wish for no more than 2 classes in a row.
  - University might want max room utilisation or minimise movement between buildings!

Evolutionaty computing – practical examples!

- Satellite dish holder boom design (Optimisation).
  - Ladder like build connects satellites body with the dish used for communication.
  - Needs to be stable, vibration resistant.
  - Keane et.al. Optimised this construction using an evolutionary algorithm
  - Resulting structure 20,000% ! Better than traditional shapes (but looks strange!)
  - There is no symmetry, no design logic visible, looks like a random drawing.
  - Shows the power of evolution!
  - Design purely driven by quality – not by preference for symmetry, aesthetics, etc.
  - This can also result in reverse engineering!
  - Once a good solution is seen, it can be analysed and explained through the lens of traditional engineering.





Evolutionaty computing – practical examples!

- Modelling of traders (Modelling)
  - Schulenburg and Ross developed trading agents using evolution.
  - Given 10 years of trading history.
  - A trader consists of rules (condition -> action)
  - Each day the condition presented to traders and action is triggered (buying or selling)
  - Periodically, a genetic algorithm is run on the rules, well performing ones are rewarded and poor ones are discarded.
  - This evolved agents which out-performed many well known trading strategies.
  - Another benefit (compared to neural networks for this task) if that rule-bases of evolved traders are easily examinable and transparent

What is an Evolutionary Algorithm

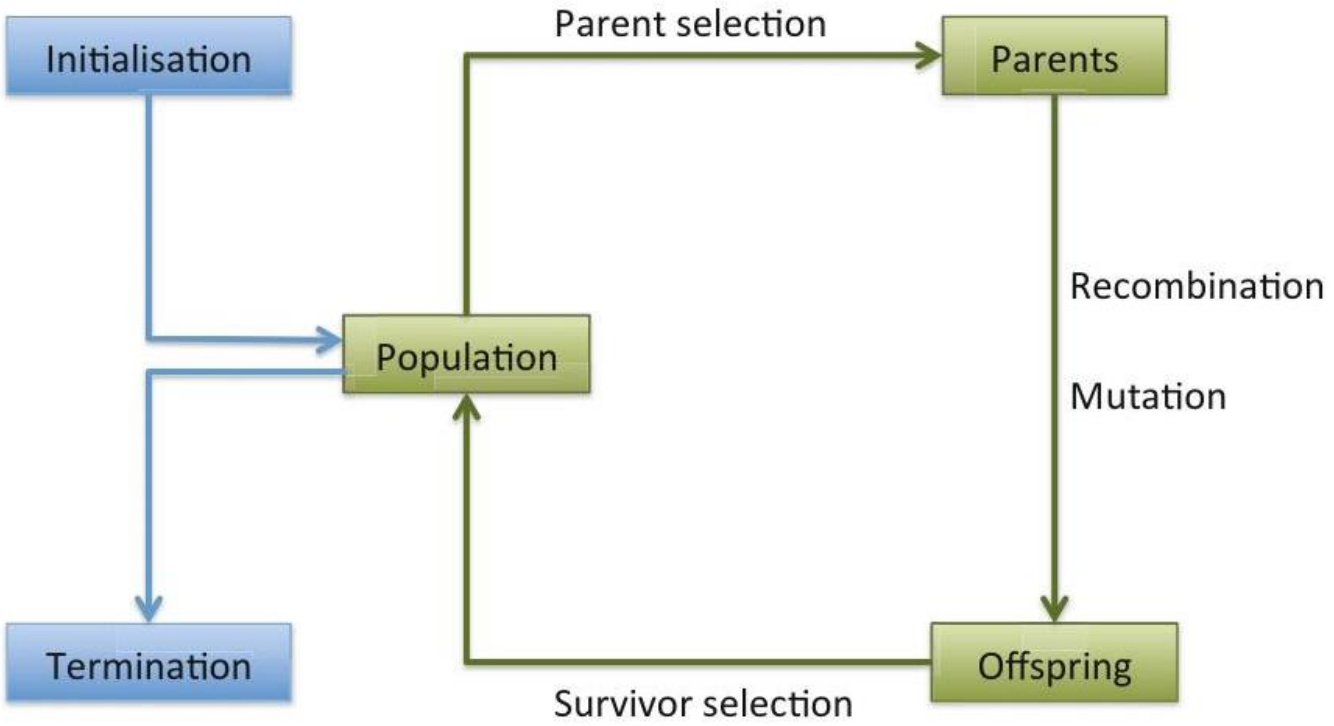
- We will study the main components of an EA.
- Their role and terminology.
- Issues related to operation of EAs

- There are many variants of EAs!
- But, common underlying idea is the same.
- Given a population in an environment with limited resources.
- Competition for resources causes selection.
- This causes rise in overall fitness of population.

What is an Evolutionary Algorithm

- We have a 'fitness function' to be maximised.
- We randomly create a set of candidate solutions.
- We apply the fitness function to the solutions to measure their fitness.
- On this basis, some of the better solutions are chosen to seed the next generation.
- This is done by applying recombination and/or mutation.
- Recombination is applied to two/more solutions (parents) producing one/more (children).
- Mutation is applied to one solution to produce a new solution.
- Applying these operators to parents results in a set of children.
- Fitness of children is evaluated.
- Children and parents compete to be placed in the next generation.
- Process repeated until a solution with sufficient quality is found or a set limit of computation reached.

What is an Evolutionary Algorithm (flowchart)



What is an Evolutionary Algorithm (pseudocode)

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Components of Evolutionary Algorithms

- Most important components -
- Representation (definition of individuals)
- Evaluation function (fitness function)
- Population
- Parent selection mechanism
- Recombination and mutation operators
- Survival selection mechanism

- To create a complete, runnable EA, we need to specify each component and define initialisation and termination conditions.

Components of Evolutionary Algorithms

- Most important components -
- **Representation (definition of individuals)**
- Link 'real world/problem' to 'EA world'.
- Often involves simplification/abstraction of the problem to create 'solutions'
- We say that in the problem context, objects that are possible solutions are called phenotypes.
- The encoding of these solutions in an EA is called genotype.
- Example: an optimisation problem where possible solutions are integers.
- A solution with value 18 is a phenotype
- If we decide to use a binary encoding then 10010 is the genotype.
- In this example, the search takes place in the genotype space.
- **encoding** and **decoding**
- Another example, for travelling salesman problem (1:London, 2:Manchester, 3:Leeds, 4: Liverpool, 5: Plymouth)
- Encoding = (1, 3, 4, 2, 5)

Components of Evolutionary Algorithms

- Most important components -
- **Evaluation Function (Fitness/Objective function)**
- Should represent the requirements the population should adapt to meet.
- Forms the basis for selection and therefore improvement.
- From the point of the problem = it is the task to be solved.
- It is the function/procedure that assigns a quality measure to the 'genotypes'
- Example = for travelling salesman problem
- Fitness function = sum (length of road between the cities in the path)

Components of Evolutionary Algorithms

- Most important components -
- **Population**
- Holds all the possible solutions.
- Given a representation, defining a population is as simple as how many individuals to hold.
- In most EAs, population size is constant.
- Selection operators work at population level.
- Example= best solution of the current population could be selected to seed next generation.
- Worst solution of current population could be selected to be replaced.
- Diversity of a population is a measure of the number of different solutions present.
- Many measures of diversity exist. (count, entropy, etc)

Components of Evolutionary Algorithms

- Most important components -
- **Parent selection**
- Role is to distinguish between individuals based on quality, to allow them to become parents for the next generation.
- An individual is a parent if it has been **selected** to undergo recombination/mutation**.**
- This is typically done, probabilistically.
- High-quality individuals have high chance of becoming parents.
- Low-quality individuals have a small but positive chance to become parents.
- This is done to prevent the population to become too greedy and get stuck in a local optimum.

Components of Evolutionary Algorithms

- Most important components -
- **Recombination**
- Merges information from two parents into one/two offspring.
- It is stochastic in nature: the choice of what parts of the parents are combined and how it is done.
- Strong supporting case on why it can be beneficial (plant/livestock breeding)
- The hope in EA is that recombination will generate some improved individuals.
- In EAs, recombination is applied probabilistically. (a small chance that it might not be applied)

Components of Evolutionary Algorithms

- Most important components -
- **Mutation**
- Applied to one individual and produces a new individual.
- It is always stochastic (random choice) in nature.

- **Replacement**
- Similar to parent selection, I.e. distinguish between individuals based on fitness.
- Used in a different stage in the EA.
- After the creation of children.
- Since population is constant, the replacement decides which individuals will be allowed in the next generation.
- It is often deterministic.
- e.g group parents and children and order by fitness. Select top p individuals for next generation

Components of Evolutionary Algorithms

- Most important components -
- **Initialisation**
- Usually kept simple in EAs.
- An initial population is generated randomly.
- Problem specific heuristics can be used to create individuals.

- **Termination**
- Max allowed CPU time.
- Total number of fitness evaluations reached.
- Fitness improvement remains under a set threshold.
- Population diversity drops under a threshold.

Concrete Examples

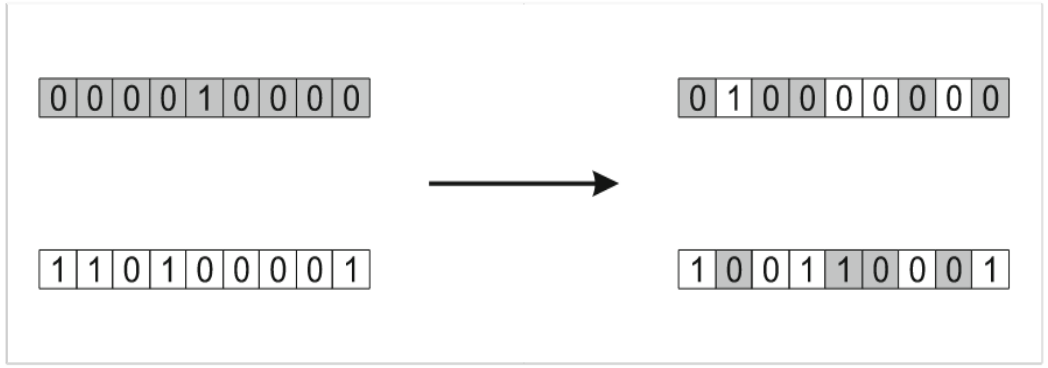
- **Binary Representation** (one of the simplest)
- Genotype = bit string of 0/1
- Natural encoding for some problems (e.g. knapsack problem) where binary decisions are involved.
- Length and decoding decided on the basis of the problem.
- We must ensure that all possible bit strings denote a valid solution.

- Mutation : bitwise/uniform

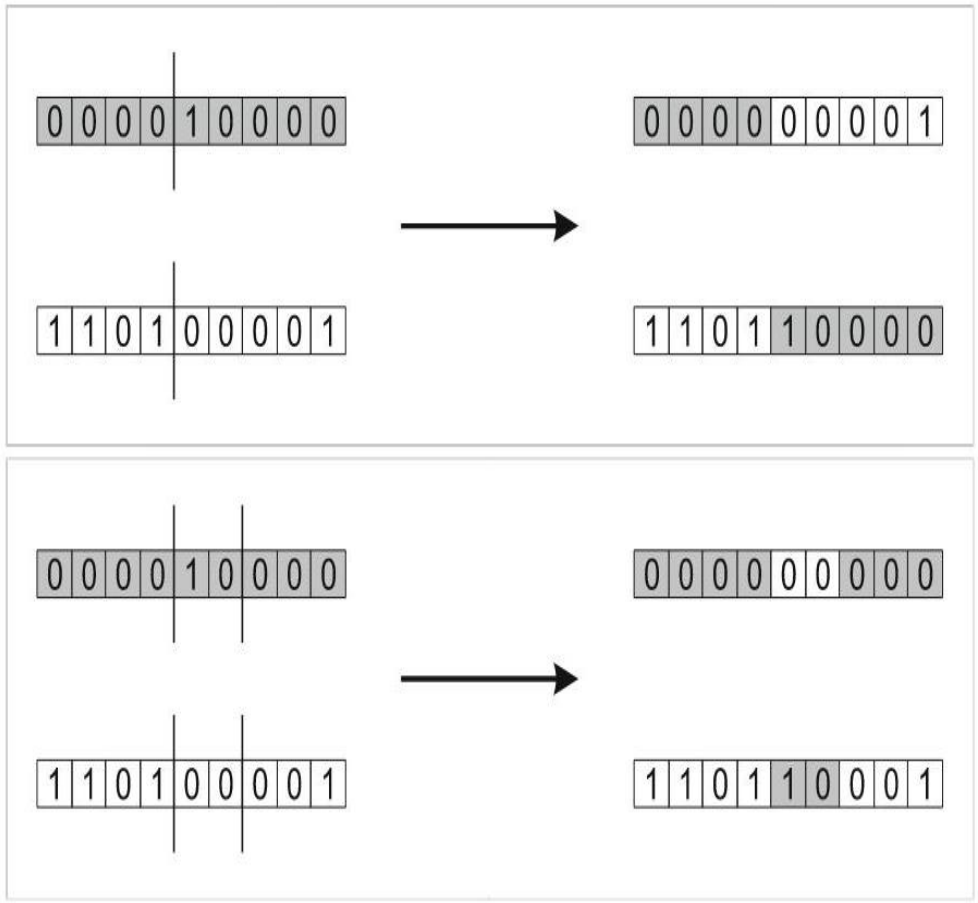
- Recombination
- One point
- Two point
- uniform

Concrete Examples

- **Binary Representation** (one of the simplest)
- Recombination
- One point
- Two point
- uniform



Uniform crossover. The array [0.3, 0.6, 0.1, 0.4, 0.8, 0.7, 0.3, 0.5, 0.3] of random numbers and $p = 0.5$ were used to decide inheritance for this example.



One-point crossover (top) and n-point crossover with $n = 2$ (bottom)

Concrete Examples

- **Integer Representation**
- As name suggests, encoding values are integers.
- An obvious examples = problem of finding the set of optimal values for a set of variables which all take integer values.
- Integers can be finite or unrestricted.
- Mutation
  - Random resetting: extension of bit flip. For each position, with probability p reset its value to a random integer.
  - 0,3,5,2,4 -> 1,3,5,7,4

  - Creep mutation: add a small value (positive or negative) to each gene with probability p.
  - The values are sampled randomly for each position.

- Recombination: Same as bit strings!

Concrete Examples

- **Real-valued Representation (Often most appropriate)**
- Values we want to represent as genes take on real values.
- Eg (they might represent length, weight, tolerance, etc)
- The genotype for a solution with k genes will be a vector <x_1, x_2,…,x_k> where each x is a real valued number.
- Mutation
  - Uniform mutation: Similar to bit flip **However** the values of each gene are drawn uniformly from [L, U] where L and U are the lower and upper bounds for that position.
  - Non uniform mutation: Similar to creep mutation. A value is drawn from gaussian distribution with mean 0 and user specified std dev, and this is added to the current gene, curtailing final value between [L, U]. It is applied with probability 1 per gene.
  - There are other types!!!

Concrete Examples

- **Real-valued representation**
- Recombination
- Simple arithmetic :

$$\text{Child 1: } \langle x_1, \ldots, x_k, \alpha \cdot y_{k+1} + (1-\alpha) \cdot x_{k+1}, \ldots, \alpha \cdot y_n + (1-\alpha) \cdot x_n \rangle.$$

- Single arithmetic :

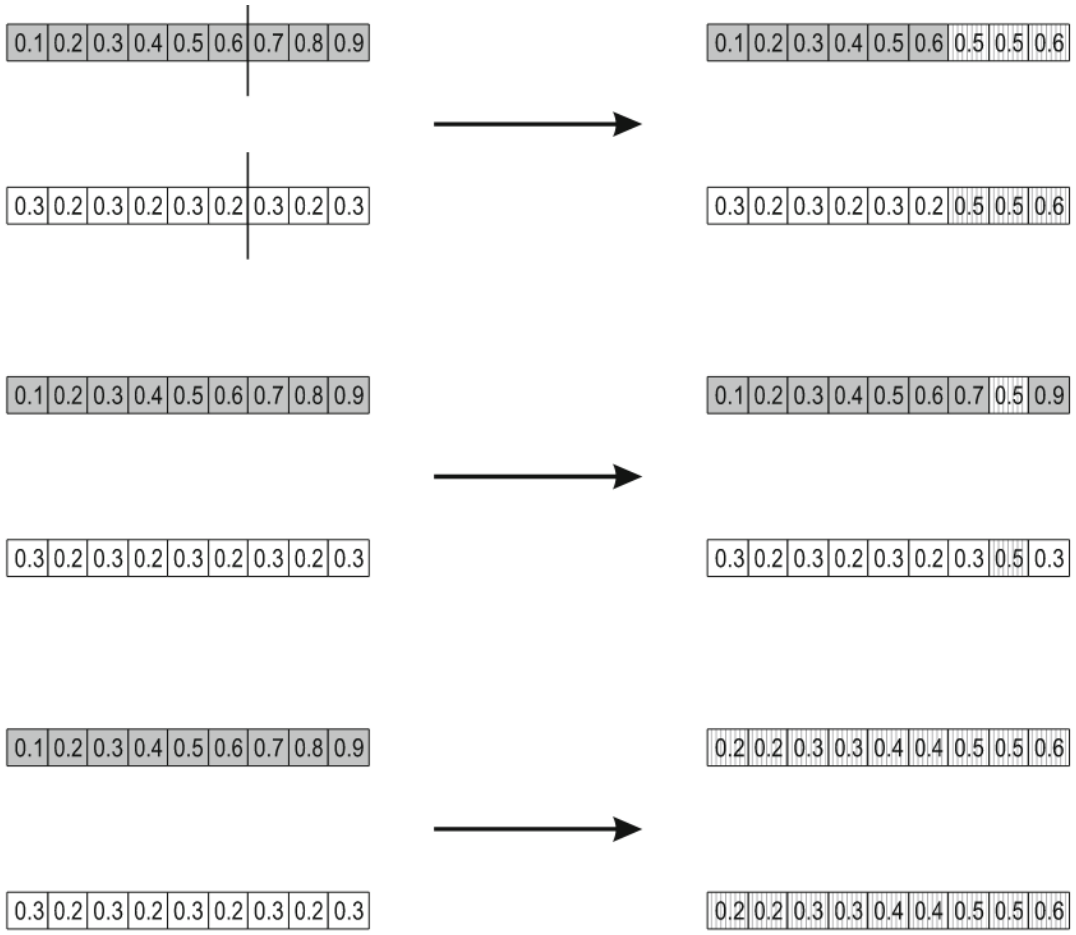
$$\text{Child 1: } \langle x_1, \ldots, x_{k-1}, \alpha \cdot y_k + (1-\alpha) \cdot x_k, x_{k+1}, \ldots, x_n \rangle.$$

- Whole arithmetic :

$$\text{Child 1} = \alpha \cdot \bar{x} + (1-\alpha) \cdot \bar{y}, \qquad \text{Child 2} = \alpha \cdot \bar{y} + (1-\alpha) \cdot \bar{x}.$$

Concrete Examples

- **Real-valued representation**
- Recombination



Simple arithmetic recombination with $k = 6, \alpha = 1/2$ (top), single arithmetic recombination with $k = 8, \alpha = 1/2$ (middle), whole arithmetic recombination with $\alpha = 1/2$ (bottom).

Concrete Examples

- **Permutation representation**
- For problems where sequence/order of events is key.
- Example - List of cities in the Travelling salesman problem.
- Mutation
  - Swap Mutation: Select two positions and swap them.

  - Insert: Select two positions, move them next to each other and shuffle others to make room.

  - Scramble: Select a region in the solution and scramble their positions.

  - Inversion: select two positions at random and reverse the order of genes between these positions.

Swap (top), insert (middle), and scramble mutation (bottom).

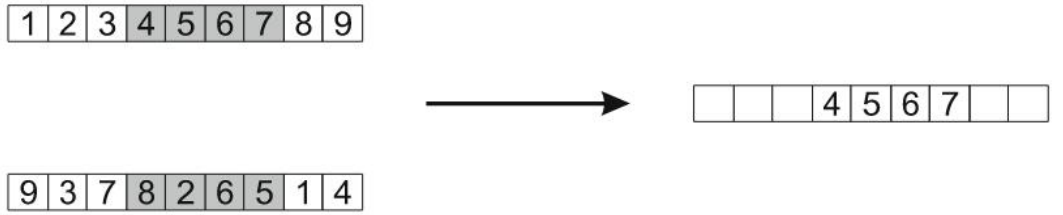
Concrete Examples

- **Permutation representation**
- For problems where sequence/order of events is key.
- Example - List of cities in the Travelling salesman problem.
- Recombination
  - Partially Mapped crossover:
    1. Choose 2 points at random, copy the segment between them from first parent (P1) to first child
    2. Starting from first crossover point look for elements in that segment of second parent (P2) that have not been copied
    3. For each of these, look at child to see what element (j) has been copied in its place from P1.
    4. Place i into position occupied by j in P2.
    5. If place occupied by j in P2 has been filled in offspring by element k, put I in position occupied by k in P2.
    6. The remaining positions are filled from P2.

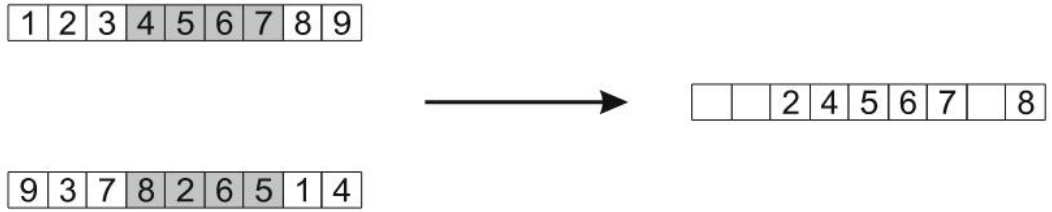
  - More types exist!

Concrete Examples

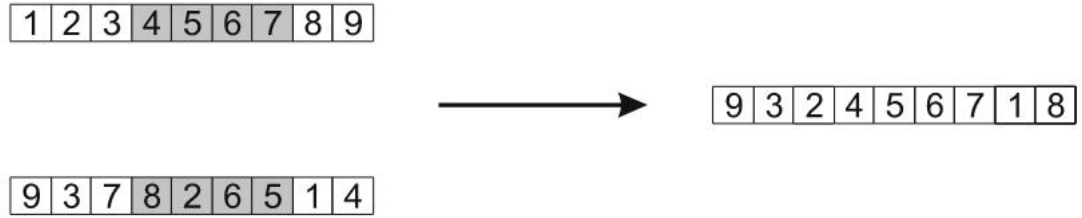
- **Permutation representation**
- PMX step 1



- PMX step 2



- PMX step 3



Concrete Examples

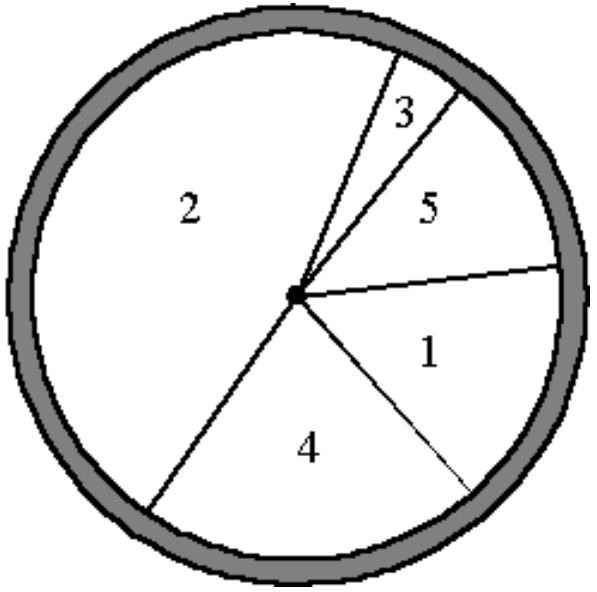
- **Population management**
- 2 different models
- Generational
  - We start with population size μ.
  - Parents selected from this.
  - From this λ children are created.
  - After each generation, μ individuals selected from offspring replace the current population.

- Steady state
  - In this case λ old individuals are replaced by new ones

Concrete Examples

- **Parent selection**
- **Fitness proportional selection**
- Probability p_i that an individual i in population P is chosen to be a parent is f(i)/**Σ**f(j)
- Also called roulette wheel selection.
- Solutions with higher fitness have more chance of being selected.

	String $\mathbf{s}_i$	Fitness $f(\mathbf{s}_i)$	Relative Fitness
$\mathbf{s}_1$	10110	2.23	0.14
$\mathbf{s}_2$	11000	7.27	0.47
$\mathbf{s}_3$	11110	1.05	0.07
$\mathbf{s}_4$	01001	3.35	0.21
$\mathbf{s}_5$	00110	1.69	0.11



Concrete Examples

- **Parent selection**
- **Tournament selection**
- Does not require global knowledge of the population.
- Simple and fast.
- K individuals are selected at random from the population.
- These are compared and the best is selected.
- This is selected as one of the selected parents.

Concrete Examples

- **Survivor selection**
- Reducing from μ parents and λ offspring to μ individuals for the next generation.

- 2 strategies are common
- Age based
  - Age = 1 in generational models
  - Can be done in steady state models, create new offspring and replace with oldest parent.
- Fitness based
  - Replace worst : replace the λ worst parents.
  - Elitism: used along with age based. Current best is always kept in the population (even if it is oldest)
  - (μ + λ): merge parents and offspring and select top μ.
  - (μ, λ): mixture of age and fitness. All parents are discarded. Offspring ranked and μ selected.

Popular variants

- **Genetic algorithms**

Representation	Bit-strings
Recombination	1-Point crossover
Mutation	Bit flip
Parent selection	Fitness proportional - implemented by Roulette Wheel
Survival selection	Generational

Table 6.1. Sketch of the simple GA

Popular variants

- **Evolution Strategies**

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	Deterministic elitist replacement by (μ, λ) or $(\mu + \lambda)$
Speciality	Self-adaptation of mutation step sizes

Sketch of ES

Popular variants

- **Evolutionary Programming**

Representation	Real-valued vectors
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic (each parent creates one offspring via mutation)
Survivor selection	Probabilistic $(\mu + \mu)$
Speciality	Self-adaptation of mutation step sizes (in meta-EP)

Sketch of EP

Measuring Performance

- **Two basic measures**

1. Solution Quality
   - Can be expressed by fitness.
2. Running speed
   - Measured in time or search effort.

- Combination of these results in three measures of algorithm performance for a single run.

1. Given a max time (number of fitness evaluations), performance is defined as best fitness at end.
2. Given a minimum fitness level. Performance is defined as time required to reach it.
3. Given a max time and min fitness, performance is defined through boolean notion of success: a run is successful if the given fitness is reached in the given time.

- Since EAs are stochastic, a good estimation of performance requires multiple runs and then taking means.
- This gives MBF (mean best fitness), AES (average evals to solution), SR (success rate) which correspond to above.

Measuring Performance

- Typical progress of EA over time

Typical progress of an EA illustrated in terms of development over time of the highest fitness in the population

Example – Knapsack problem

- **0-1 knapsack problem**

Example – Knapsack problem

- **0-1 knapsack problem**
- Given a set of n items, each has some value and some cost.
- Task is to select a subset of those items that maximises the sum of the values
- While keeping the summed cost within some max capacity C.
- Representation :
- We use a bit string of length n, where 0 means item is not selected and 1 means it is.
- HOWEVER, we keep a tally of cost from left to right and an item with 1 is included if it does not exceed capacity.
- Therefore, every possible string represents a valid solution!

Example – Knapsack problem

- **0-1 knapsack problem**
- Representation :
- We use a bit string of length n, where 0 means item is not selected and 1 means it is.
- HOWEVER, we keep a tally of cost from left to right and an item with 1 is included if it does not exceed capacity.
- 5 items (example from figure): 0: (4$, 12kg), 1 : (2$, 2kg), 2:(1$,1kg), 3:(10$,4kg), 4:(2$, kg1)
- C = 15kg
- Possible solution : 00110 = 1$, 1kg + 10$, 4kg = 11$, 5kg (valid)
- Another one: 11110 = 12kg+2kg+1kg+1kg > 15 so, we only include first 3 items = 15 kg, 7$
- Can use one of the bit string mutation here and one of the recombination operators.