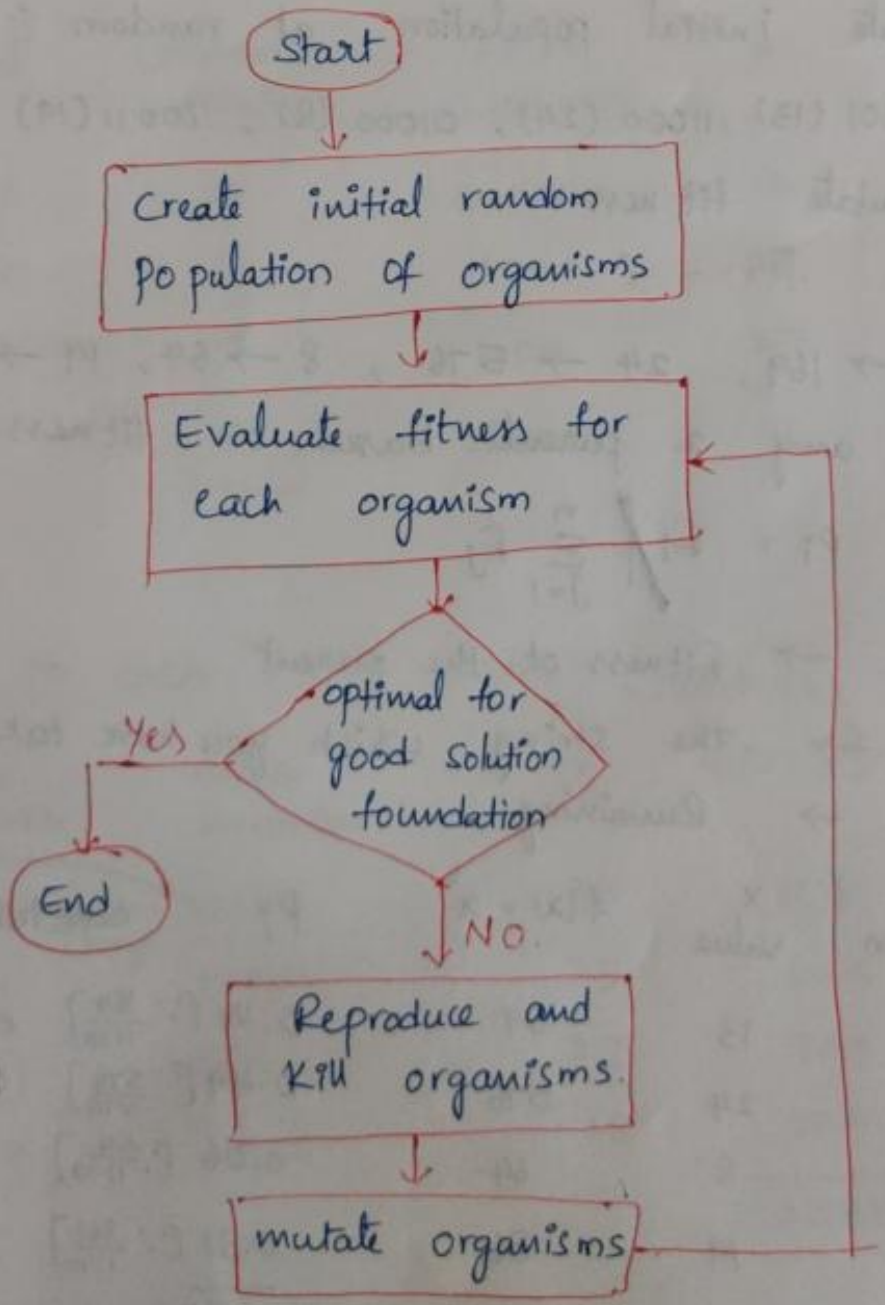
mood-book

* Genetic Algorithms :-

- They belong to evolutionary Algorithms
  - Adaptive heuristic (Short cuts) search Algorithm
  - Based on genetics and natural selection
  - Used to generate high quality solution for an optimisation problem.

* Flow chart :-



* Operations of Genetic Algorithm :

1) Selection
2) Crossover
3) Mutation
4) Encoding

* Example for Genetic Algorithm :

→ $f(x) = x^2$, maximize this function with 'x' in interval [0, 31].

(1.) Generate initial population at random (Called genotypes)

01101 (13), 11000 (24), 01000 (8), 10011 (19) [No. of organisms N = 4]

(2.) Calculate fitness

$f(x) = x^2$

13 → 169, 24 → 576, 8 → 64, 19 → 361

(3.) Select any 2 parents based on fitness

$$P_i = F_i / \sum_{j=1}^{n} F_j$$

P_i → Fitness of the parent

i → The string which you have taken (13/24/8/19)

J → Remaining

NO	Initial Population	X value	$f(x) = x^2$	P_i	expected count = N x P. N = 4
1	01101	13	169	0.14 [$\because \frac{169}{1170}$]	0.56 [$\because$ 4 x 0.14]
2	11000	24	576	0.49 [$\because \frac{576}{1170}$]	1.97 [$\because$ 4 x 0.49]
3	01000	8	64	0.06 [$\because \frac{64}{1170}$]	0.22 [$\because$ 4 x 0.06] → The smallest value will be replaced with 1.97
4	10011	19	361	0.31 [$\because \frac{361}{1170}$]	1.23
			1170	1.00	4.00

→ Parent (having max. value = 1.97) selected. JV ②

④. Crossover

- Can be either one point / 2 point / n point

ex:- Let us take data as

```
10011101          10001011
10101011    ⇒     10111101
```

] one point cross over

If it is two point cross over.

```
100|111|01          10001001
101|010|11    ⇒     10111111
```

NO.	Initial	Crossover point	After Crossover (16 8 4 2 1)	x	$f(x) = x^2$
1	01101	4	01100	12	144
2	11000	4	11001	25	625 (no change in mutation)
3	11000 (01000→ replaced with 11000)	2	11011	27	729
4	10011	2	10000	16	256
					1754

3 bcoz of 0.22 less value

⑤. Mutation

Applied to each child after crossover.

NO.	After Crossover	After mutation	x	$f(x) = x^2$
1	01100 (random change)	11100	26	676
2	11001	11001	25	625
3	11011	11011	27	729
4	10000	10100	18	324
				2354

* Genetic Programming :

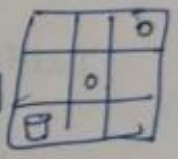
- Extension of genetic algorithms.
- Main idea - represent a Computer program as tree - used when exact solution is not known in advance

(all operations are same as genetic algorithm) - Selection, crossover, mutation & encoding

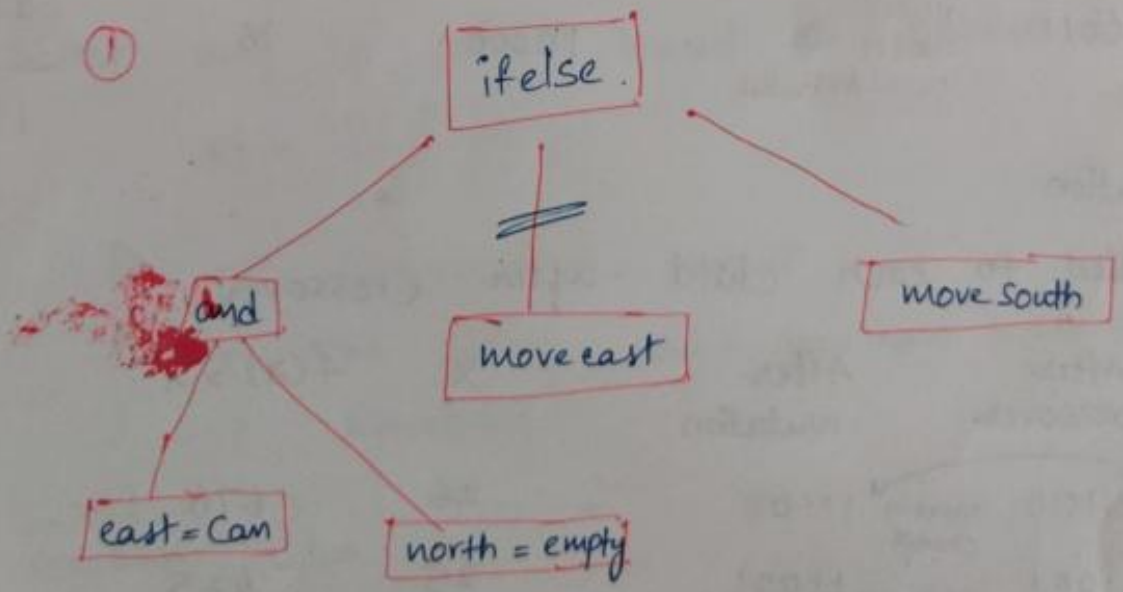
Ex: Plastic can collecting robot navigation.

```
if (east = can & north = empty)
        then move east
        else move south
```

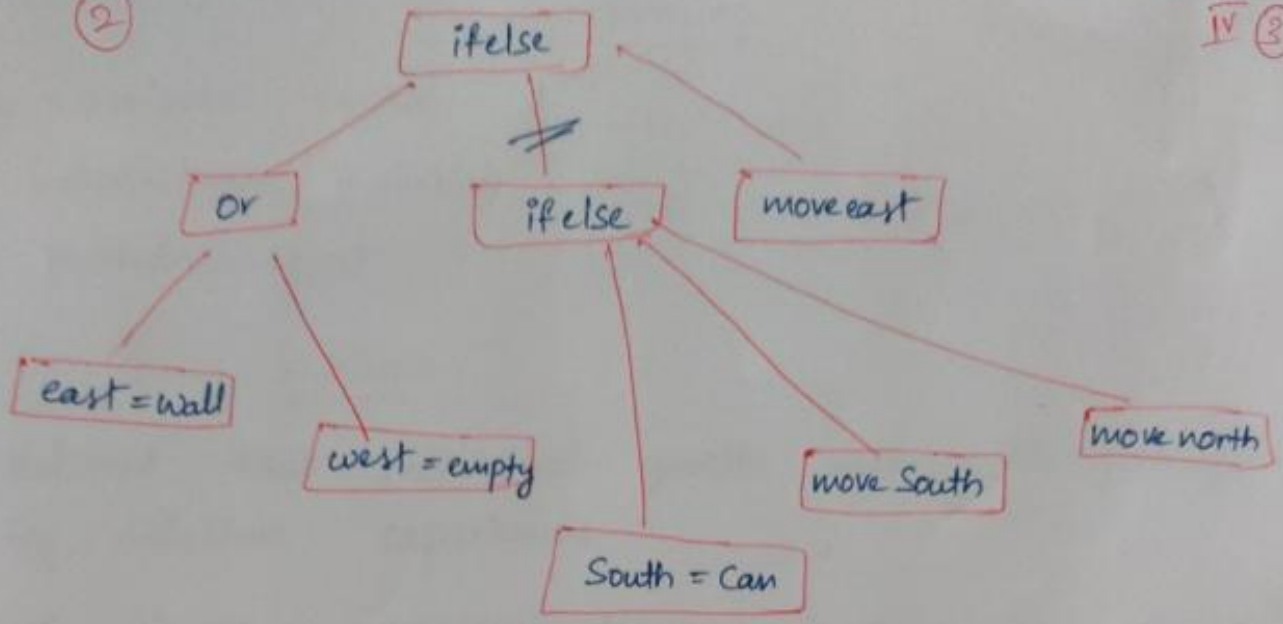
[∵ North is empty]



→ The main concept is we have to build a tree of our Computer program.

① 

(2)



IV (3)

→ In order to make these two trees more efficient, we have to do the crossover operation

* Models of Evolution & Learning.

2 Evolution models.

1) Lamarckian evolution
2) Baldwin effect

* Lamarckian Evolution :

→ Believed that individual genetic makeup is changed by lifetime experiences.

i.e if an organism changes during its life to ~~adopt to~~ in order to adopt to the environment, then those changes are passed to its offsprings

Example : Giraffe.

* Baldwin Effect :

→ Baldwin explained about the learning behaviour of the organisms.

→ 2 things considered

1. Genotype - Genetic Code (DNA)
   - global search
2. phenotype - Your Characteristics (behaviour)
   - local search

→ Measures Cost of learning not interms of money but interms of time and energy.

Example : Fish.

* Parallelising Genetic Algorithms :-

→ Parallel genetic algorithm uses multiple algorithms to solve a single task

2 Categories

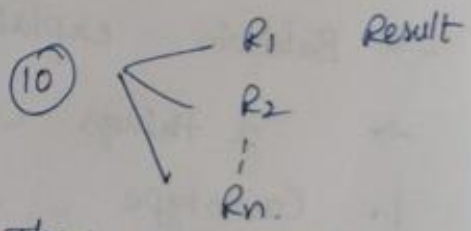
1. Fine - Grained :

Detailed description which deals with much smaller components.

2. Coarse - Grained :

Divides into fewer components (Size of component is more than that of fine - grained)

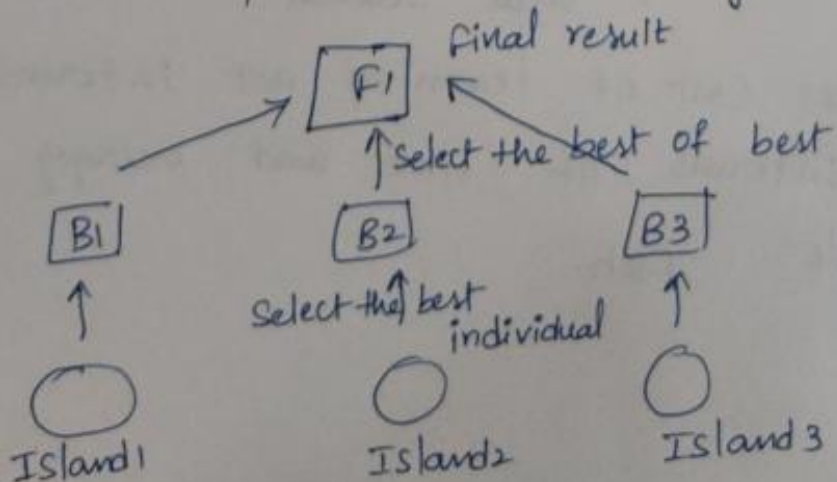
→ All the algorithms solve the same task and once they obtain solution, best one is selected.



→ Also called as island model

→ They do not depend on CPU : They can run parallelly

Advantage : Avoids problem of crowding.



* Learning Set of rules:-

2 ways

① way → First learn decision tree and translate that tree into rules, one rule for each leaf node.

② way → Use a genetic algorithm that encodes each rule as a bit string.

→ we have 2 types of algorithms that learn directly from set of rules.

1. First order rules
2. Sequential Covering algorithm

* Sequential Covering Algorithm:-

— It sequentially covers each and every rule
— Extracts rules from dataset directly.

It is mainly based on one of the evaluation measure — 1. Accuracy.
2. Coverage

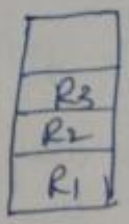
* Algorithm:

1. It creates an empty Set of decision list (R)
2. A function called "learn one rule" function is used.
   it extracts best rule for class "y"

If all training records ∈ class y ⇒ +

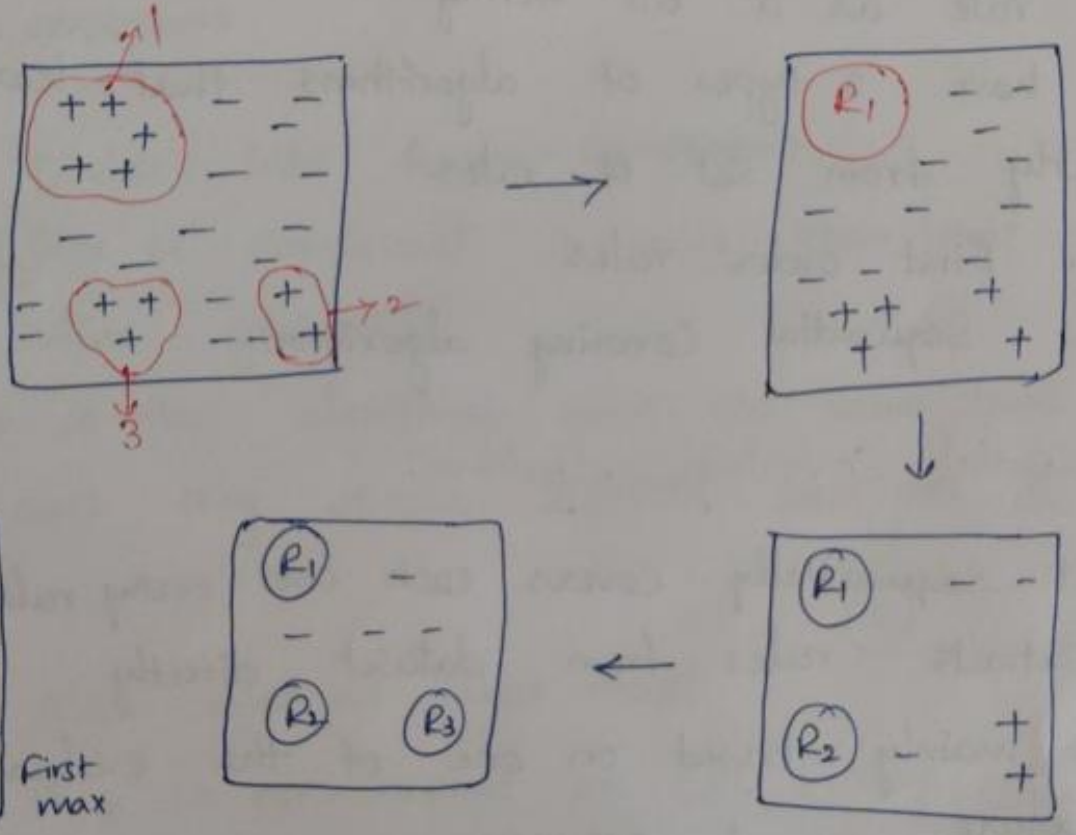
If all training records of class y ⇒ –

3. Get desirable values (only +ve)
4. Eliminate records (i.e the desirable ones)
5. New rule is added to bottom of R



Example :

Let us take a dataset with both +ve & –ve samples



* First order Rule learning :-

- implemented with the help of PROLOG (uses horn clauses)
- First order logic is much expressive than Propositional logic
  - It allows a finer grain of specification & reasoning

Example : daughter (x, y)

where (x, y) are {Name, Mother, Father, Male, Female}

Training examples are represented as :

(person 1, person 2, target attribute)

P1 ((name 1 = Ann, mother 1 = Mary, Father 1 = Bob, male 1 = F, Female 1 = T)

P2 (name 2 = bob, mother 2 = gill, Father 2 = Joe, male 2 = T, Female 2 = F)

Target → daughter (1, 2) = T

→ with this example → only few rules

if (father 1 = Bob) ∧ (Name 2 = Bob) ∧ (Female 1 = T)

then daughter 1, 2 = T

From first order rule,

If father (y, x) ∧ Female (y) then daughter (x, y)

Terms in expressions :

Constants → Bob, 23

Variables → x, y, z

Predicate Symbols → Female, Father (only T/F)

Function Symbols → age (only const

Connectivities → (∧, ∨, →, ←)

Quantifiers → (∀, ∃) [shows the quantity whole part, some part]

* FOIL Algorithm :- (First order Inductive Learning)

- Extension of Sequential learning (Covering)

* 1 Similarity - 1 rule at a time
* 1 Difference - only when target literal is true then only it will learn

FOIL searches its hypothesis by using 2 nested loops

* 1. Outer loop : disjunctive (∨) OR
* 2. Inner loop : Conjunction (∧) AND

(from most general to more specific).

→ Performance of FOIL Algorithm :

$$FOIL\_Gain(L, R) = t\left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0}\right)$$

L - Candidate literal that is to be added to R

p_0 - No. of +ve bindings of R [R - rule set]

n_0 - No. of -ve " " R. [L - Candidate literal planning to add into R]

p_1 - " " +ve " " (R+L) (After adding L to R)

n_1 - " " -ve " " (R+L)

t - " " +ve " " R also coved by (R+L)

* REINFORCEMENT LEARNING :-

→ Learns depending on changes occuring in environment.

Goal : To achieve the best result.

Example 1 : chessboard

Goal : To win the game.

Example 2 : Agent

(2 ways fire and water)

fire - wrong choice. (-50 points)

water - Correct choice (+50 points)

whenever, choice is made (right or wrong) feedback is to agent.

Feedback - Based on this right choice is taken.

* Markov decision problem :

Examples :

Q- learning

Temporal Difference

* Q - LEARNING (Example of Reinforcement learning)

Q - Quality.

Terms Required :

1. Policy : $Q^{\pi}(S_t, a_t)$ Ex: Maze game

certain rules & limits

S - state, A - Action, t - time.

→ Policy is undertaken by agent

↓

(he observes the given state and selects best possible action)

Example: Empty road - Speed driving.

Crowdy road - Slow driving.

2. Reward: Scalar Quantity. (magnitude)

Ex:- Maze game.

Correct action → reward.

3. Penalty:- Commonly called as -ve reward

wrong action.

Ex: Parking in no parking zone.

→ Q-learning mainly depends on ② factors.

1. Q-Function
2. Q-Table.

* Q-Function:

$$Q^{\pi}(S_t, a_t) = E(R_{t+1} + \gamma R_{t+2} + \cdots \gamma^n R_{t+n} | S_t a_t)$$

↓

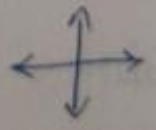
Bellman Equation

γ - Discount factor

R - Reward
S - state
a - action
t - time

* Q-Table:

→ Combination of actions & states.

Example: In a game ⇒ 

Actions: Up, down, Right, left

State: Start, end, Reward, health etc.

Steps followed :

1) Exploration : Explore (finding) all possible paths.

2). Exploitation : Best possible path is identified

Remove duplicates 3) Initialise Q table → All values '0'

4) Choose Action

5) Perform Action

6) Measure reward.

7) Update Q - table

	↑	→	↓	←
Start				
reward				
Health				
END				

Initially all values = 0

* Temporal Difference Learning :

→ Takes advantage of both monte carlo (mc) ideas and dynamic programming (DP)

- Combination of mc and DP

* monte carlo Ideas (mc)

learns directly from raw experience i.e without model

- There is no predefined model

* Dynamic Programming (DP) :

→ It estimates based on part of learning rather than waiting for the final outcome.

→ Relation between TD, Dp & mc methods are always cyclic.

more about temporal difference Learning :

- It is a model free learning.
- It has 2 important properties.

1. It does not require the model to be known in advance
2. It also can be applied for non-episodic tasks.

- Temporal difference learning (Agent's current action effects the future action) uses an update rule (Sequential) for updating the value of a state.

$$V(S_t') \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

$V(S_t')$ ← updated value

$V(S_t)$ – value of previous state

α – learning rate (or) step size

γ – discount factor

R – Reward.

$V(S_{t+1})$ – Value of current state