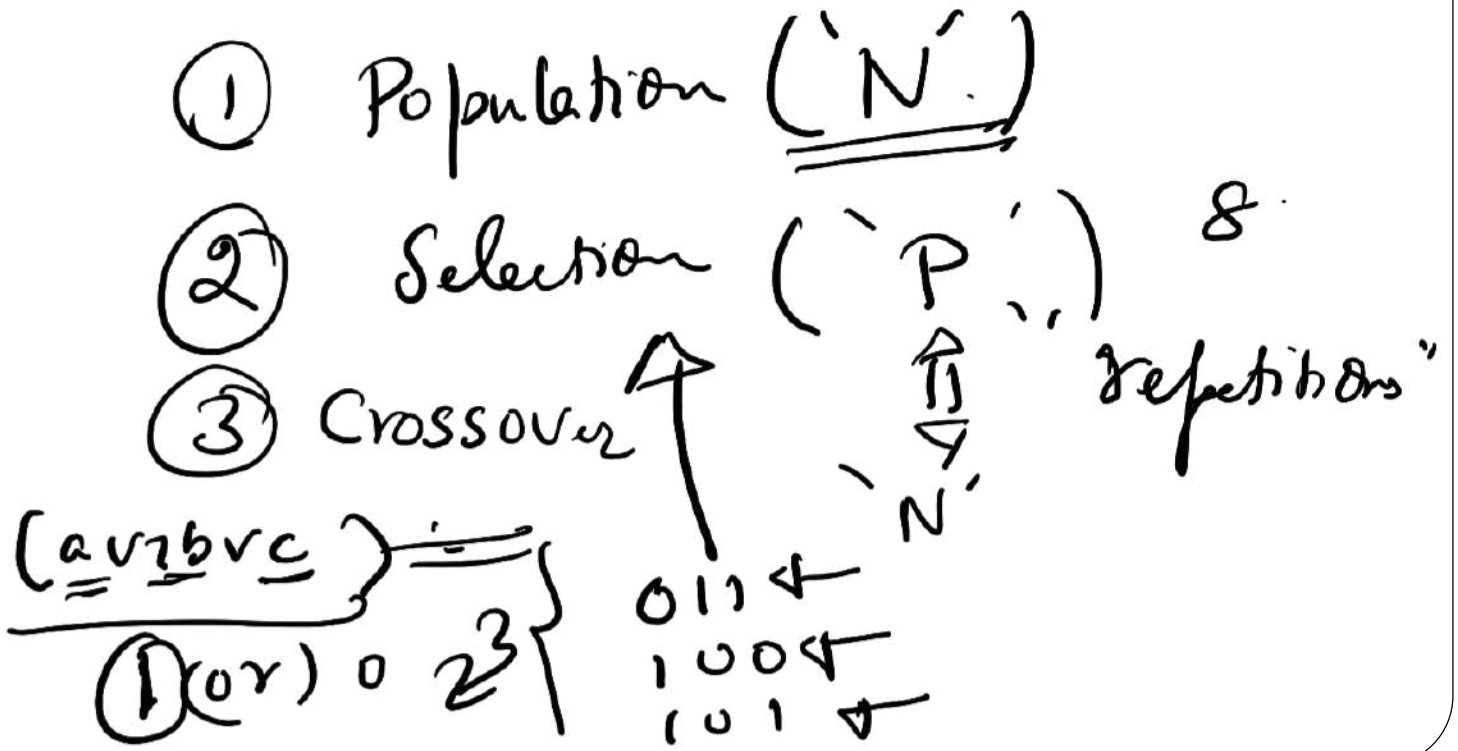
Population based Methods :-

* Evolved by nature.
* N Candidate Solutions ← Start



GENETIC-ALGORITHM()

```
1  P ← create N candidate solutions        ▷ initial population
2  repeat
3      compute fitness value for each member of P
4      S ← with probability proportional to fitness value,
           randomly select N members from P
5      offspring ← partition S into two halves, and randomly mate
                   and crossover members to generate N offsprings
6      with a low probability mutate some offsprings
7      replace k weakest members of P with k strongest offsprings
8  until some termination criteria
9  return the best member of P
```

P. $f(x)$ ← Square of 5 bit string.

Initial	Binary	f(x)	Prob.	Expected	Actual
01101	13	169	0.14	0.58	1
11000	24	576	0.49	1.97	2
01000	8	64	0.06	0.22	0
10011	19	361	0.31	1.23	1

Total 1170

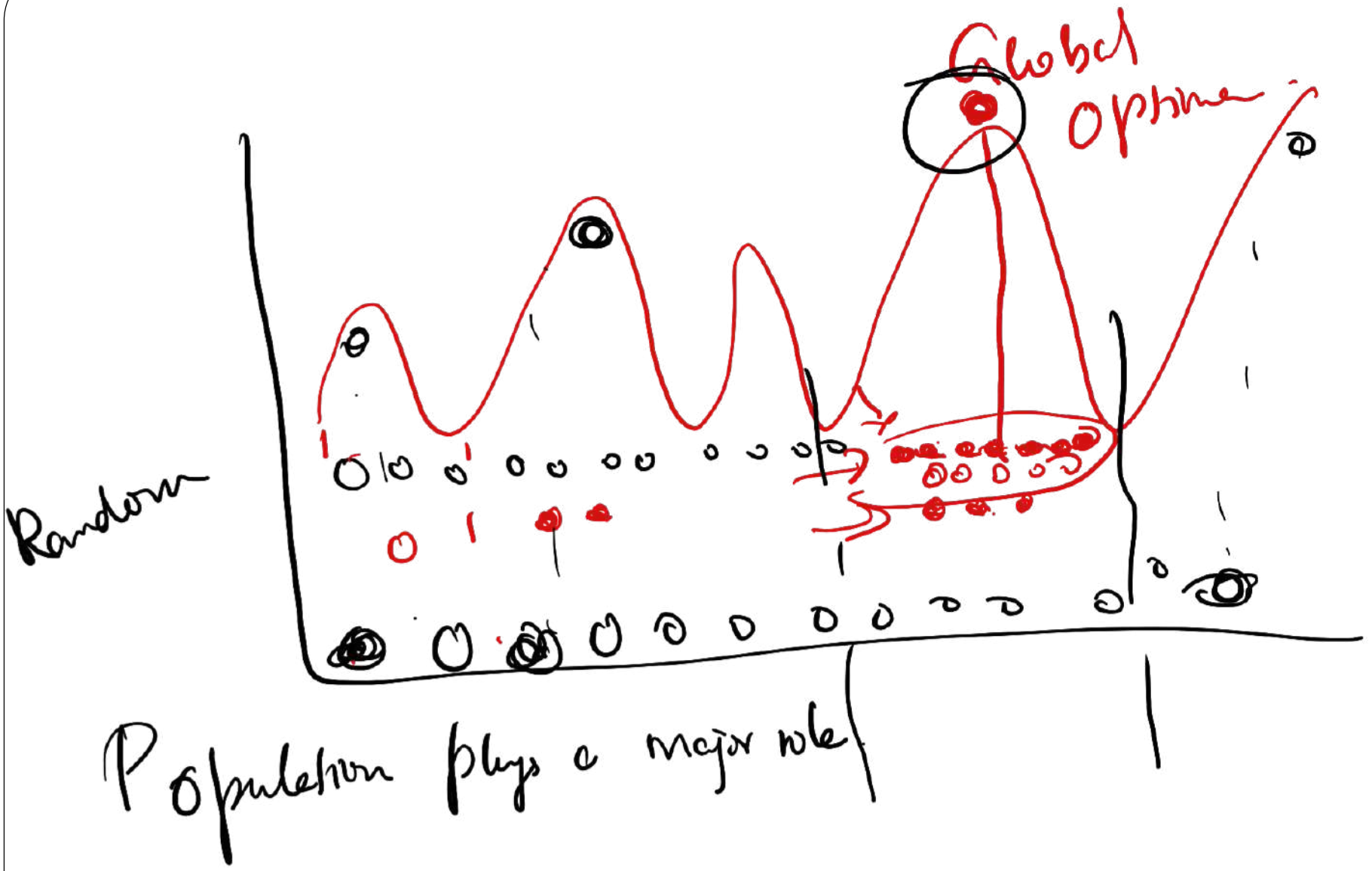
S

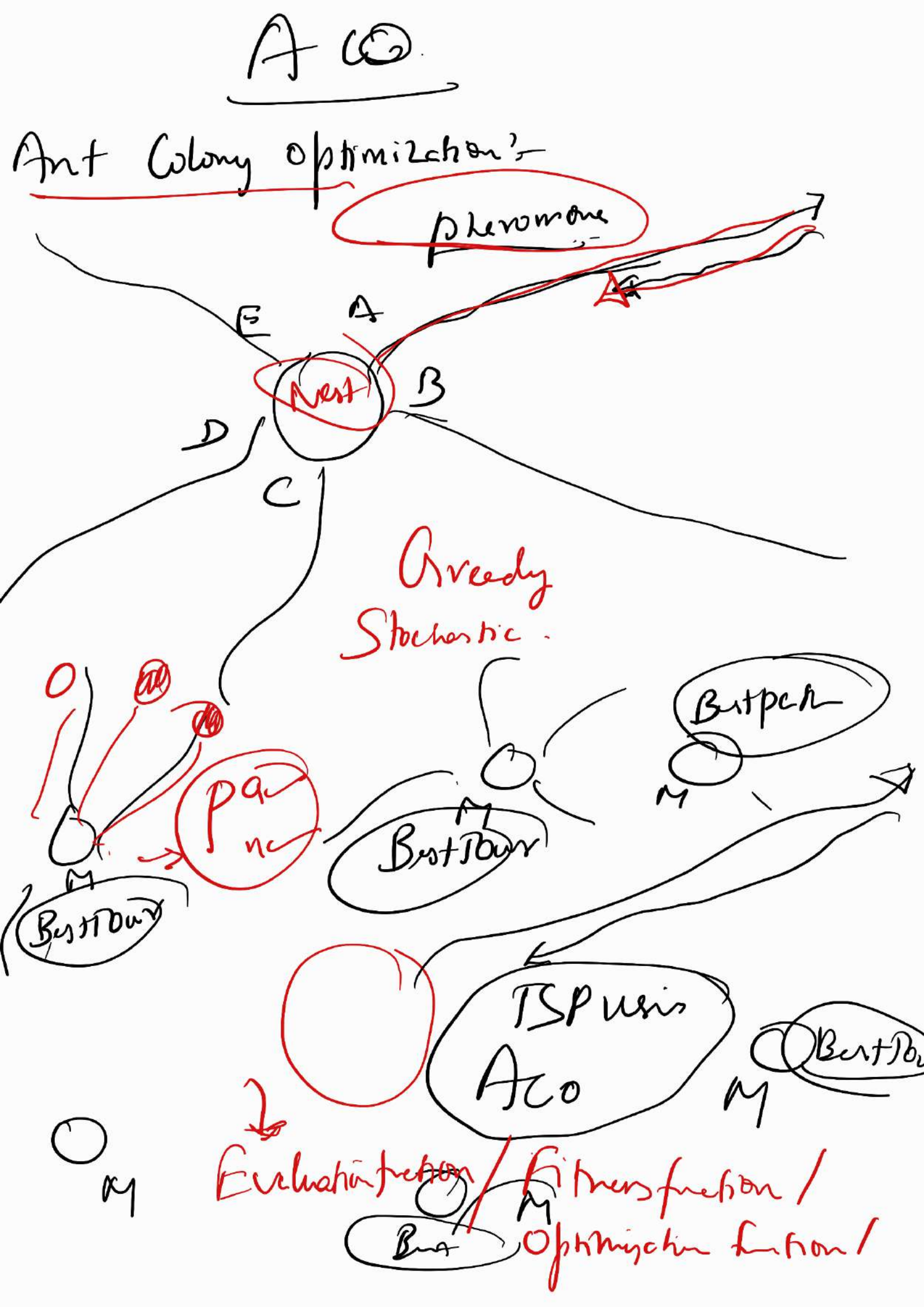
11000
11000
01101
10011

11000 11000
01101 10011

11001 11011
01100 10000

Crossover ⇒ offspring.


Global optima
Random
Population plays a major role


ACO
Ant Colony optimization
Pheromone
E
A
Nest
B
D
C
Greedy
Stochastic
Best Tour
TSP using
ACO
Evaluation function / Fitness function /
Objective function /

Adversarial Search:-

① In a multiagent environment {regard as game}, the impact of each agent on others is significant.

② All the agents are rational, deterministic.

③ The utility values at end of each game are always equal and opposite. → payoff {+1, -1, 0} {W, L, D}.

④ The state of a game is easy to represent, and agents are restricted to a small number of actions.

A game can be formally defined as:

① Initial State ← Common for both agents.

② Player(s) ← Alternate moves will be given to the players.

Actions(s) ←

Result(s,a) ←

Terminal_Test(s) ← +1, -1, 0

Utility Function – Evaluation function / Pay off

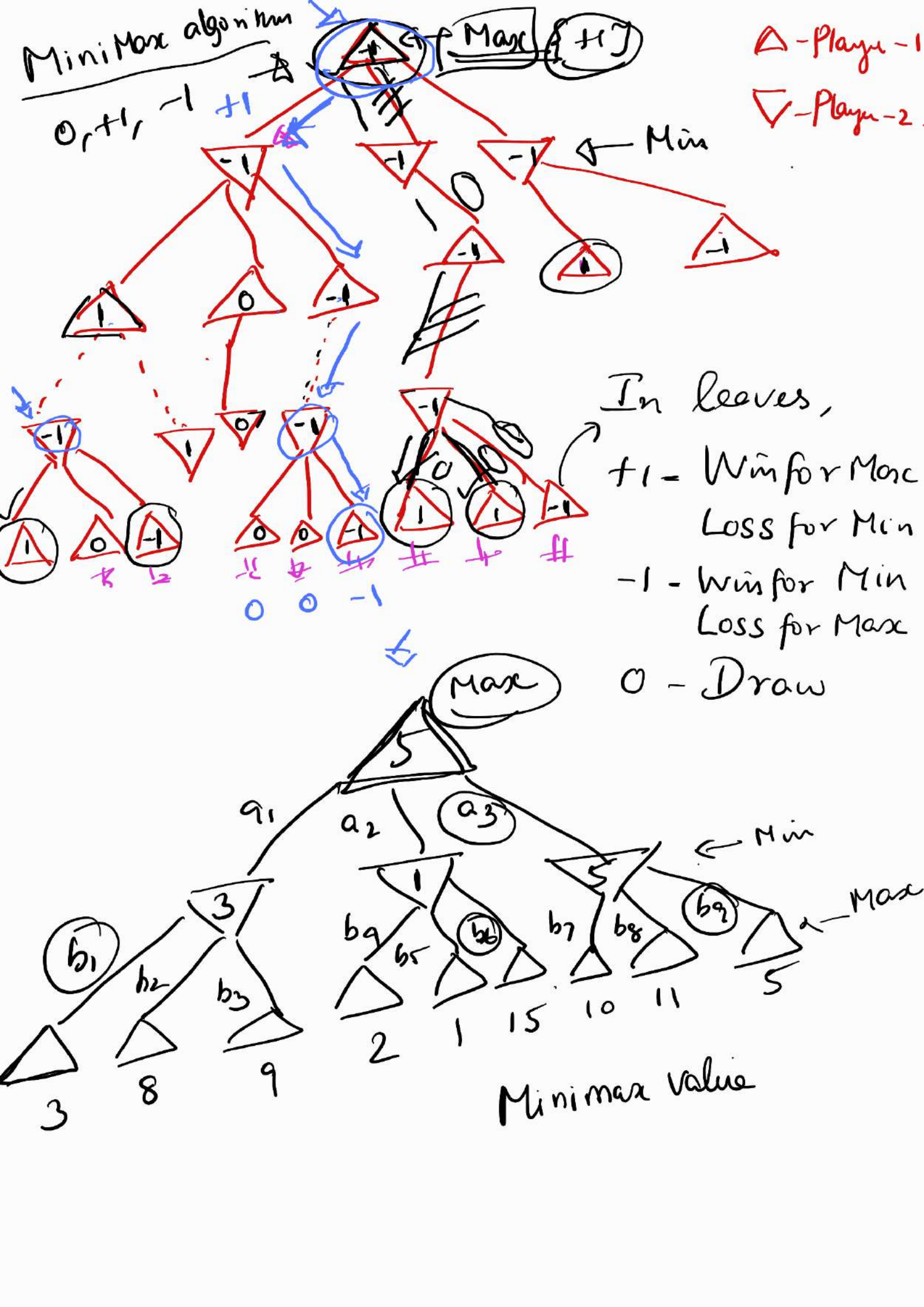
– Find numeric value in a game that ends in terminal state 's' for a player.

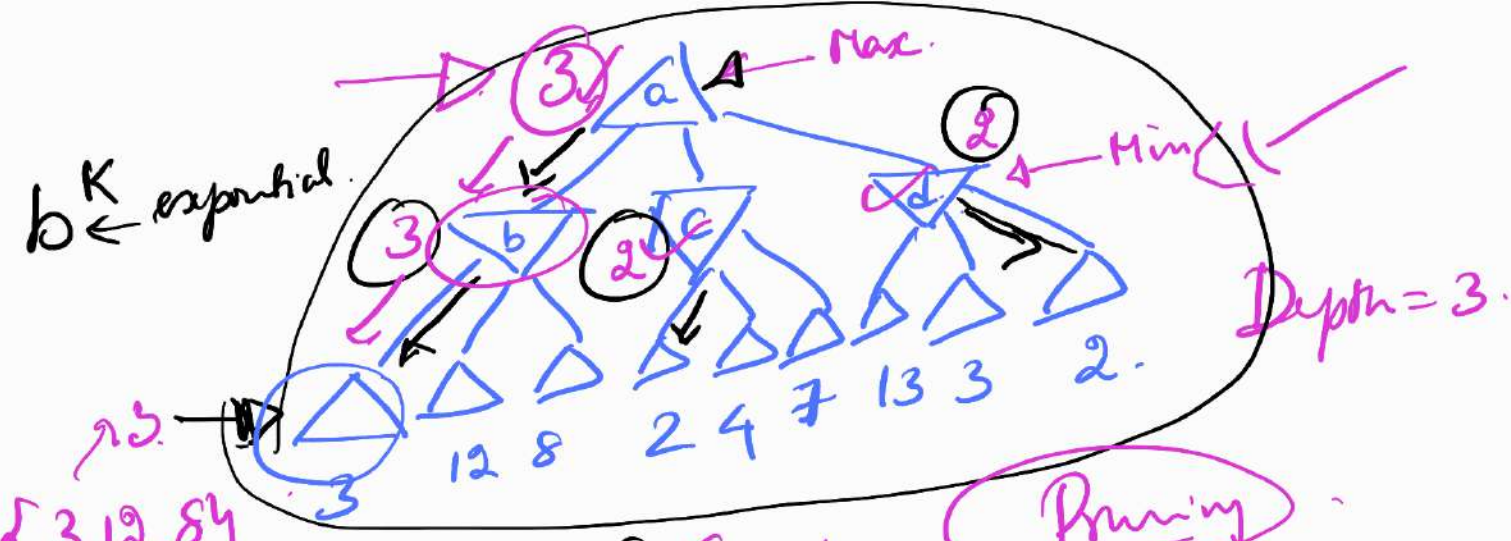
Zero-Sum Game:-

Utility (s, a) ← payoff

→ After several movement of states, in a certain instance of game, the utility function would be.

$$\left\{ \begin{matrix} 0+1 \\ 1+0 \\ \frac{1}{2}+\frac{1}{2} \end{matrix} \right\} = 1$$


MiniMax algorithm
Max
Min
△ - Player - 1
▽ - Player - 2
0, +1, -1
+1
In leaves,
+1 - Win for Max
Loss for Min
-1 - Win for Min
Loss for Max
0 - Draw
Max
Min
Max
Minimax value



b^K ← exponential.

{3, 12, 8}

Minimax backup rule.

{ Greedy.
  Depth first exploration.

Pruning.

Complexity:- Branching factor → b (3). b^K ← Time.
depth → K. b ← Space

K - depth tree (K - ply Tree) :-

Minimax (S).

If B is a terminal node. // Base Case

return h(n) {+1, -1, 0}

Else if B is a Max node.

Value ← -∞

for each child C in B.

Value ← max (value, Minimax(c))

Else if B is a Min node.

Value → ∞

for each child C in B.

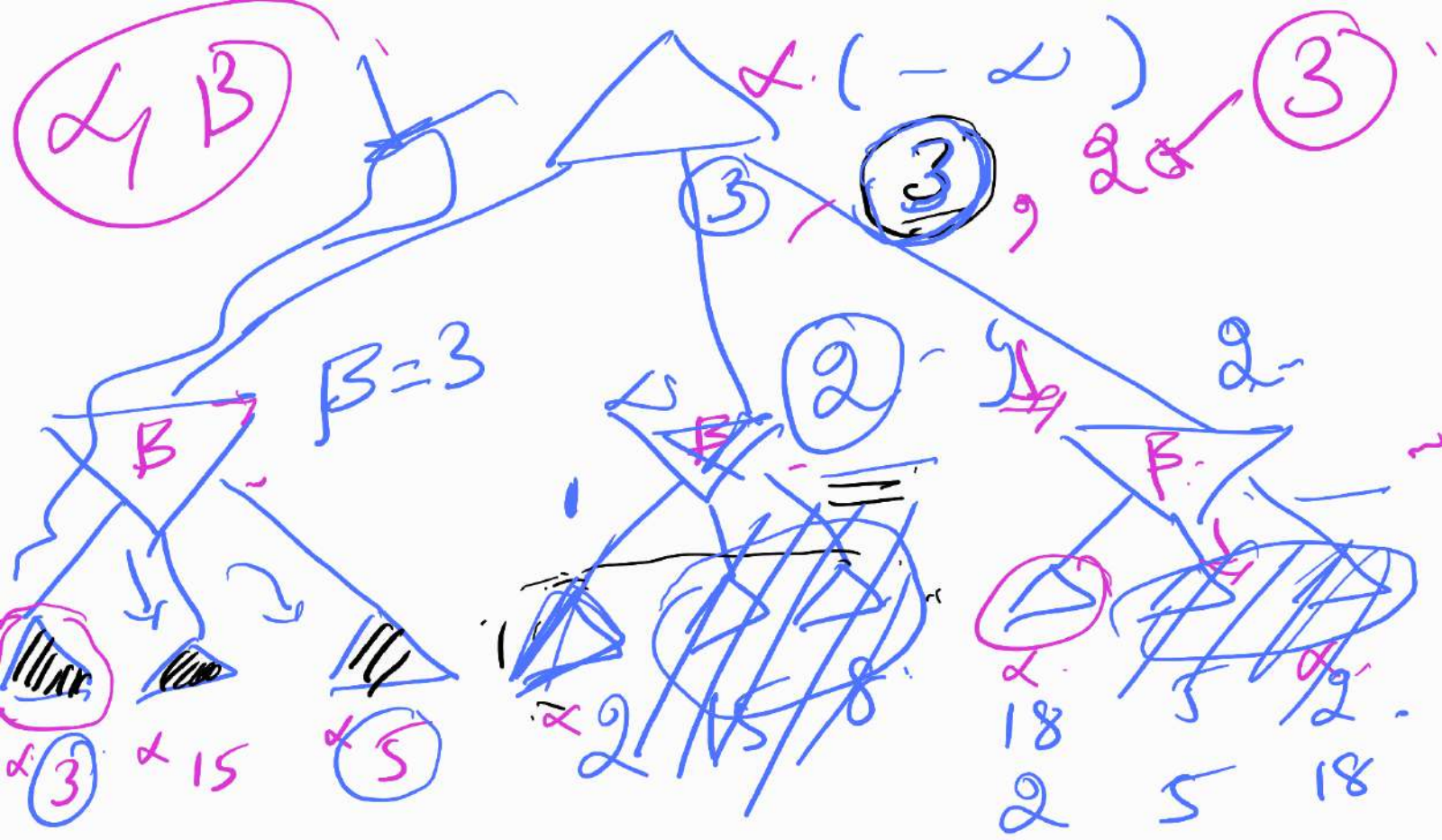
Value ← min (value, minimax(c))

Alpha-Beta Pruning :-

Alpha → Max nodes.

Beta → Min nodes.

Minimax Backup



$3 < \infty$

$15 < 3$

$5 < 3$

$2 < \infty$

$18 < \infty$

$5 < 18$

$2 < 5$

Alpha: Alpha is a value assigned to the max node, through the path towards the goal, that gets updated if any greater value is found than the existing value

* min

* lower

* For Beta)