21CS2213RA
AI for Data Science

Session -10

Contents: Genetic Algorithm

(DEEMED TO BE UNIVERSITY)

Genetic Algorithm

- Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GAs are a subset of a much larger branch of computation known as **Evolutionary Computation**.

Basic Terminology

- **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **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.

Examples include the different genes or sets of genes in a DNA which are responsible for different traits like blue eyes, dark hair, fair complexion etc.

- **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.

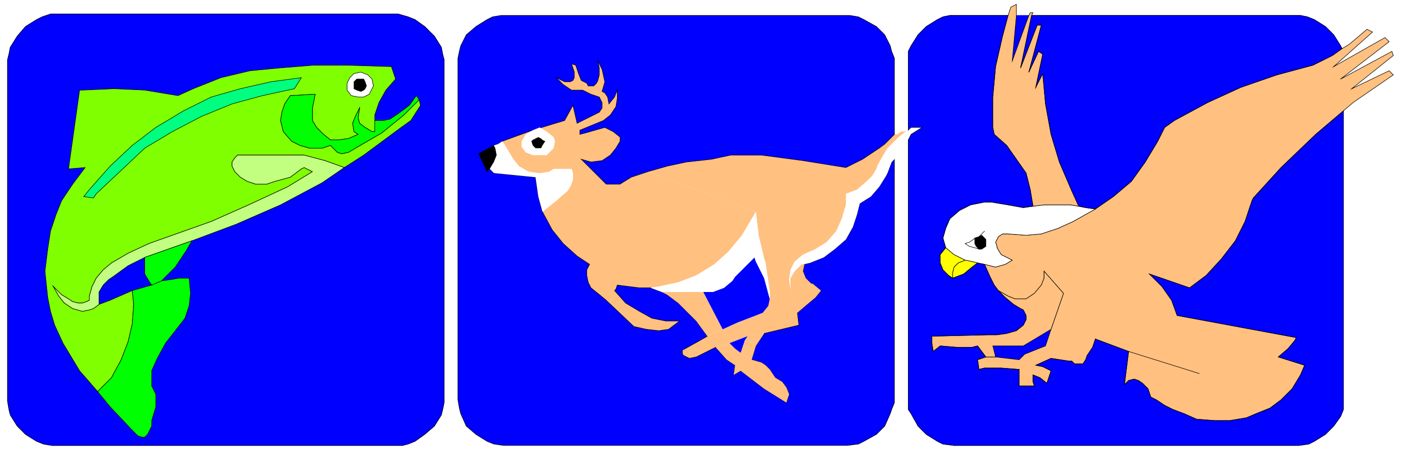
Examples include eye colour, hair color, skin colour, behavior etc.

- **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.

Simple Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

A Simple Example



"The Gene is by far the most sophisticated program around."

- Bill Gates, *Business Week*, June 27, 1994

A Simple 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) Dunedin	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 [redacted])

Child (5 8 7 ~~2 1 6 3 4)~~

This operator is called the *Order1* crossover.

Mutation

Mutation involves reordering of the list:

```
                  *       *
Before:     (5  8  7  2  1  6  3  4)

After:      (5  8  6  2  1  7  3  4)
```

Issues for GA Practitioners

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators
- Termination Criteria
- Performance, scalability
- Solution is only as good as the evaluation function (often hardest part)

Online Search Agents and Unknown Environment

- Offline search algorithms compute a complete solution before setting foot in the real world and then execute the solution. In ONLINE SEARCH contrast, an **online search agent interleaves computation and action: first it takes an action, then it observes the environment and computes the next action.**

- Online search is a good idea in dynamic or semi-dynamic domains—domains where there is a penalty for sitting around and computing too long.

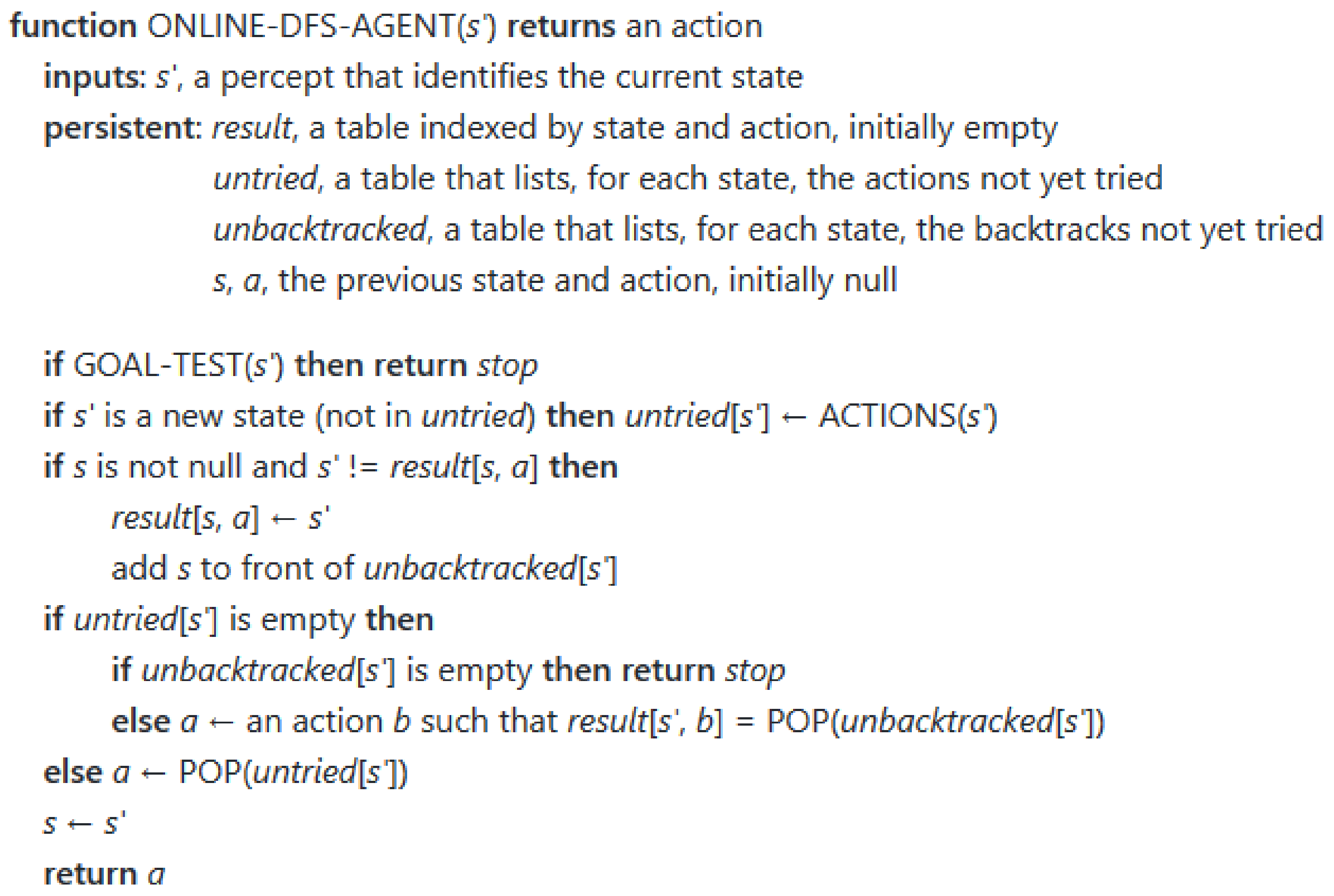
- Online search is also helpful in nondeterministic domains because it allows the agent to focus its computational efforts on the contingencies that actually arise rather than those that might happen but probably won't.
- Online search is a necessary idea for unknown environments, where the agent does not know what states exist or what its actions do. In this state of ignorance, the agent faces an **exploration problem** and must use its actions as experiments in order to learn enough to make deliberation worthwhile.

An agent knows only the following in online search:

- **ACTIONS(s)**, which returns a list of actions allowed in state s;
- **The step-cost function c(s, a, s')—**note that this cannot be used until the agent knows that s' is the outcome; and
- **GOAL-TEST(s**).

1. An online search problem must be solved by an agent executing actions, rather than by pure computation.
2. An agent cannot determine RESULT(s, a) except by actually being in s and doing a.
3. Competitive Ratio should be small and not moves to infinite.

Online DFS Agent



```
function ONLINE-DFS-AGENT(s') returns an action
  inputs: s', a percept that identifies the current state
  persistent: result, a table indexed by state and action, initially empty
              untried, a table that lists, for each state, the actions not yet tried
              unbacktracked, a table that lists, for each state, the backtracks not yet tried
              s, a, the previous state and action, initially null

  if GOAL-TEST(s') then return stop
  if s' is a new state (not in untried) then untried[s'] $\leftarrow$ ACTIONS(s')
  if s is not null and s' != result[s, a] then
      result[s, a] $\leftarrow$ s'
      add s to front of unbacktracked[s']
  if untried[s'] is empty then
      if unbacktracked[s'] is empty then return stop
      else a $\leftarrow$ an action b such that result[s', b] = POP(unbacktracked[s'])
  else a $\leftarrow$ POP(untried[s'])
  s $\leftarrow$ s'
  return a
```

End of Session

Thank you

(DEEMED TO BE UNIVERSITY)