Introduction to Genetic Algorithms

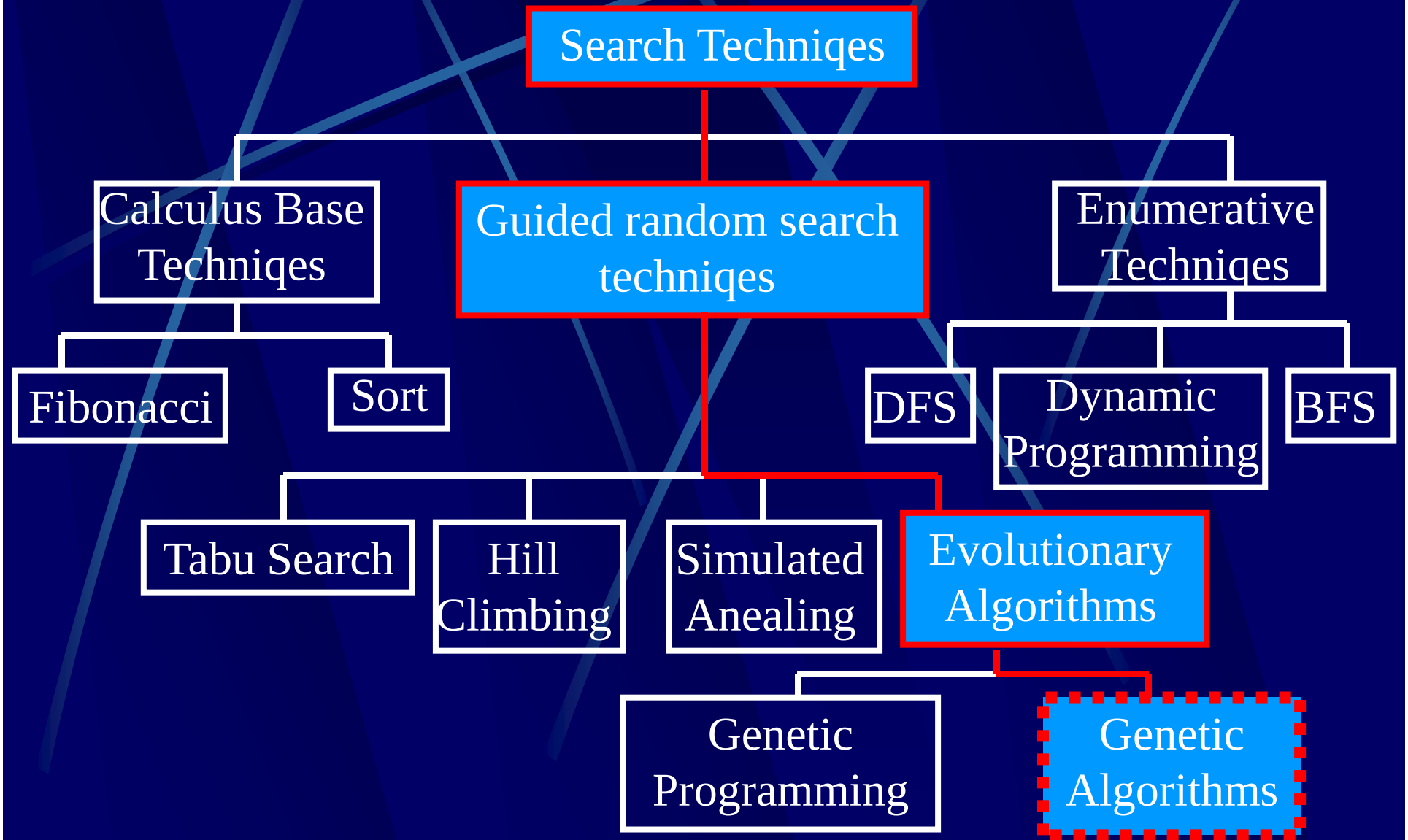
Genetic Algorithms (GA) OVERVIEW

- A class of probabilistic optimization algorithms
- Inspired by the biological evolution process
- Uses concepts of “Natural Selection” and “Genetic Inheritance” (Darwin 1859)
- Originally developed by John Holland (1975)

GA overview (cont)

- Particularly well suited for hard problems where little is known about the underlying search space
- Widely-used in business, science and engineering

Classes of Search Techniques



A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators

Stochastic operators

- ***Selection*** replicates the most successful solutions found in a population at a rate proportional to their relative quality
- ***Recombination*** decomposes two distinct solutions and then randomly mixes their parts to form novel solutions
- Crossover – new offspring
- ***Mutation*** randomly perturbs a candidate solution

The Metaphor

Genetic Algorithm	Nature
Optimization problem	Environment
Feasible solutions	Individuals living in that environment
Solutions quality (fitness function)	Individual's degree of adaptation to its surrounding environment

The Metaphor (cont)

Genetic Algorithm	Nature
A set of feasible solutions	A population of organisms (species)
Stochastic operators	Selection, recombination and mutation in nature's evolutionary process
Iteratively applying a set of stochastic operators on a set of feasible solutions	Evolution of populations to suit their environment

Simple Genetic Algorithm

produce an initial population of individuals

evaluate the fitness of all individuals

while termination condition not met **do**

    select fitter individuals for reproduction

    recombine between individuals

    mutate individuals

    evaluate the fitness of the modified individuals

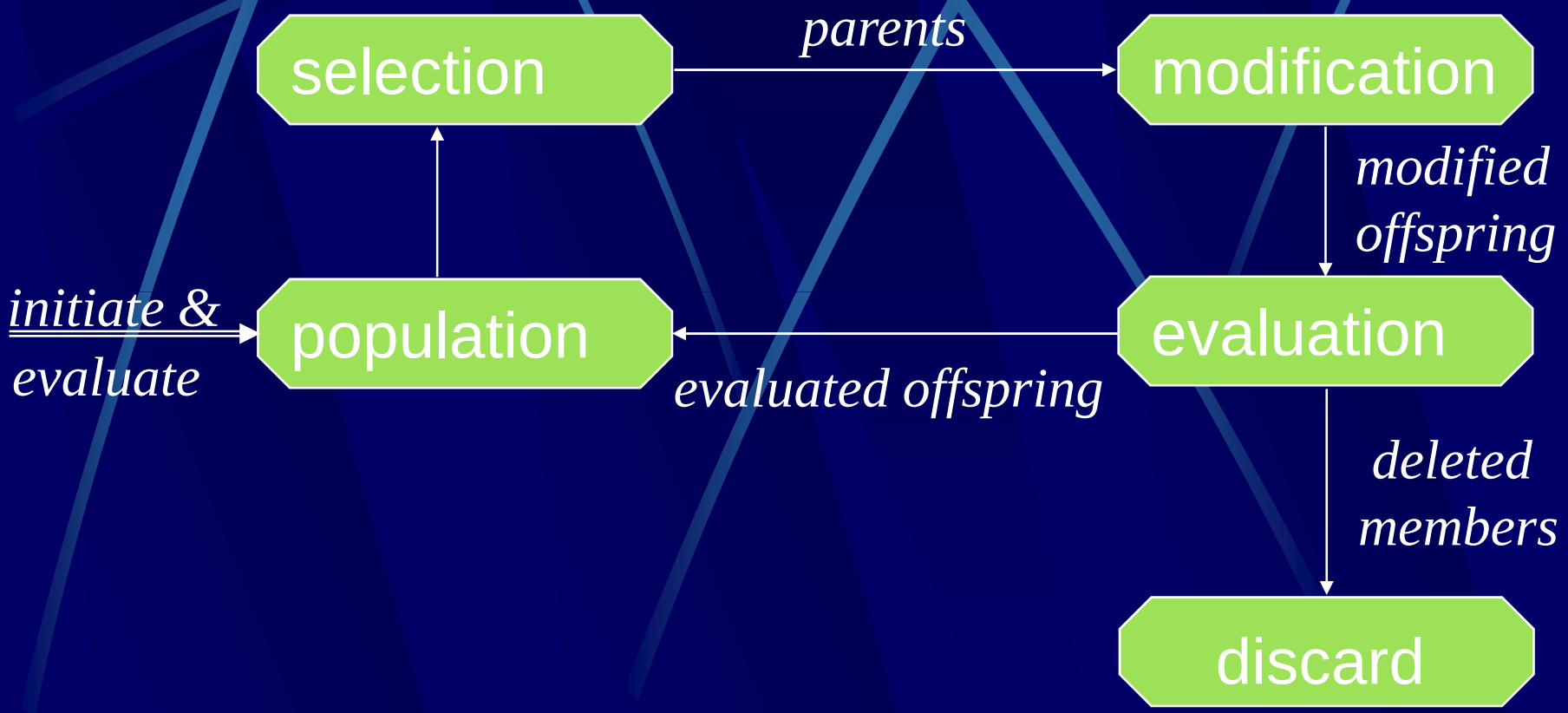
    generate a new population

End while

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
    inputs: population, a set of individuals
            FITNESS-FN, a function that measures the fitness of an individual
    repeat
        new_population ← empty set
        loop for i from 1 to SIZE(population) do
            x ← RANDOM-SELECTION(population, FITNESS-FN)
            y ← RANDOM-SELECTION (population, FITNESS-FN)
            child ← REPRODUCE(x, y)
            if (small random probability) then child ← MUTATE(child)
            add child to new_population
        population ← new_population
    until some individual is fit enough, or enough time has elapsed
    return the best individual in population, according to FITNESS-FN
```

```
function REPRODUCE(x, y) returns an individual
    inputs: x, y, parent individuals
n ← LENGTH(x)
c ← random number from 1 to n
return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c+1, n))
```

The Evolutionary Cycle



Components of a GA

A problem definition as input, and

- Encoding principles (gene, chromosome)
- Initialization procedure (creation)
- Selection of parents (reproduction)
- Genetic operators (mutation, recombination)
- Evaluation function (environment)
- Termination condition

Representation (encoding)

Possible individual's encoding

- 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 (cont)

When choosing an encoding method rely on the following key ideas

- Use a data structure as close as possible to the natural representation
- Write appropriate genetic operators as needed
- If possible, ensure that all genotypes correspond to feasible solutions
- If possible, ensure that genetic operators preserve feasibility

Initialization

Start with a population of randomly generated individuals, or use

- A previously saved population
- A set of solutions provided by a human expert
- A set of solutions provided by another heuristic algorithm

Selection

- Purpose: to focus the search in promising regions of the space
- Inspiration: Darwin's "survival of the fittest"
- Trade-off between *exploration* and *exploitation* of the search space

Next we shall discuss possible selection methods

Fitness Proportionate Selection

- Derived by Holland as the optimal trade-off between exploration and exploitation

Drawbacks

- Different selection for $f_1(x)$ *and* $f_2(x) = f_1(x) + c$
- *Superindividuals* cause convergence (that may be premature)

Linear Ranking Selection

Based on sorting of individuals by decreasing fitness

The probability to be extracted for the ith individual in the ranking is defined as

$$p(i) = \frac{1}{n}\left[\beta - 2(\beta - 1)\frac{i-1}{n-1}\right], 1 \le \beta \le 2$$

where β can be interpreted as the expected sampling rate of the best individual

Local Tournament Selection

Extracts k individuals from the population with uniform probability (without re-insertion) and makes them play a “tournament”, where the probability for an individual to win is generally proportional to its fitness

Selection pressure is directly proportional to the number k of participants

Recombination (Crossover)

* Enables the evolutionary process to move toward promising regions of the search space

* Matches good parents' sub-solutions to construct better offspring

Mutation

Purpose: to simulate the effect of errors that happen with low probability during duplication

Results:

- Movement in the search space
- Restoration of lost information to the population

Evaluation (fitness function)

- Solution is only as good as the evaluation function; choosing a good one is often the hardest part
- Similar-encoded solutions should have a similar fitness

Termination condition

Examples:

- A pre-determined number of generations or time has elapsed
- A satisfactory solution has been achieved
- No improvement in solution quality has taken place for a pre-determined number of generations

Applications

Graph coloring problem

TSP

Time table - Scheduling problem

The Traveling Salesman Problem (TSP)

The traveling salesman must visit every city in his territory exactly once and then return to the starting point; given the cost of travel between all cities, how should he plan his itinerary for minimum total cost of the entire tour?

TSP $\in$ NP-Complete

Note: we shall discuss a single possible approach to approximate the TSP by GAs

TSP (Representation, Evaluation, Initialization and Selection)

A vector $v = (i_1\, i_2 \ldots\, i_n)$ represents a tour (v is a permutation of {1,2,…,n})

Fitness f of a solution is the inverse cost of the corresponding tour

Initialization: use either some heuristics, or a random sample of permutations of {1,2,…,n}

We shall use the fitness proportionate selection

TSP (Crossover1)

OX – builds offspring by choosing a sub-sequence of a tour from one parent and preserving the relative order of cities from the other parent and feasibility

Example:

p_1 = (1 2 3 4 5 6 7 8 9) and

p_2 = (4 5 2 1 8 7 6 9 3)

First, the segments between cut points are copied into offspring

o_1 = (x x x 4 5 6 7 x x) and

o_2 = (x x x 1 8 7 6 x x)

TSP (Crossover2)

Next, starting from the second cut point of one parent, the cities from the other parent are copied in the same order

The sequence of the cities in the second parent is

9 – 3 – 4 – 5 – 2 – 1 – 8 – 7 – 6

After removal of cities from the first offspring we get

9 – 3 – 2 – 1 – 8

This sequence is placed in the first offspring

o_1 = (2 1 8 4 5 6 7 9 3), and similarly in the second

o_2 = (3 4 5 1 8 7 6 9 2)

TSP (Inversion)

The sub-string between two randomly selected points in the path is reversed

Example:

(1 2 3 4 5 6 7 8 9) is changed into (1 2 7 6 5 4 3 8 9)

Such simple inversion guarantees that the resulting offspring is a legal tour