Introduction to Genetic Algorithms

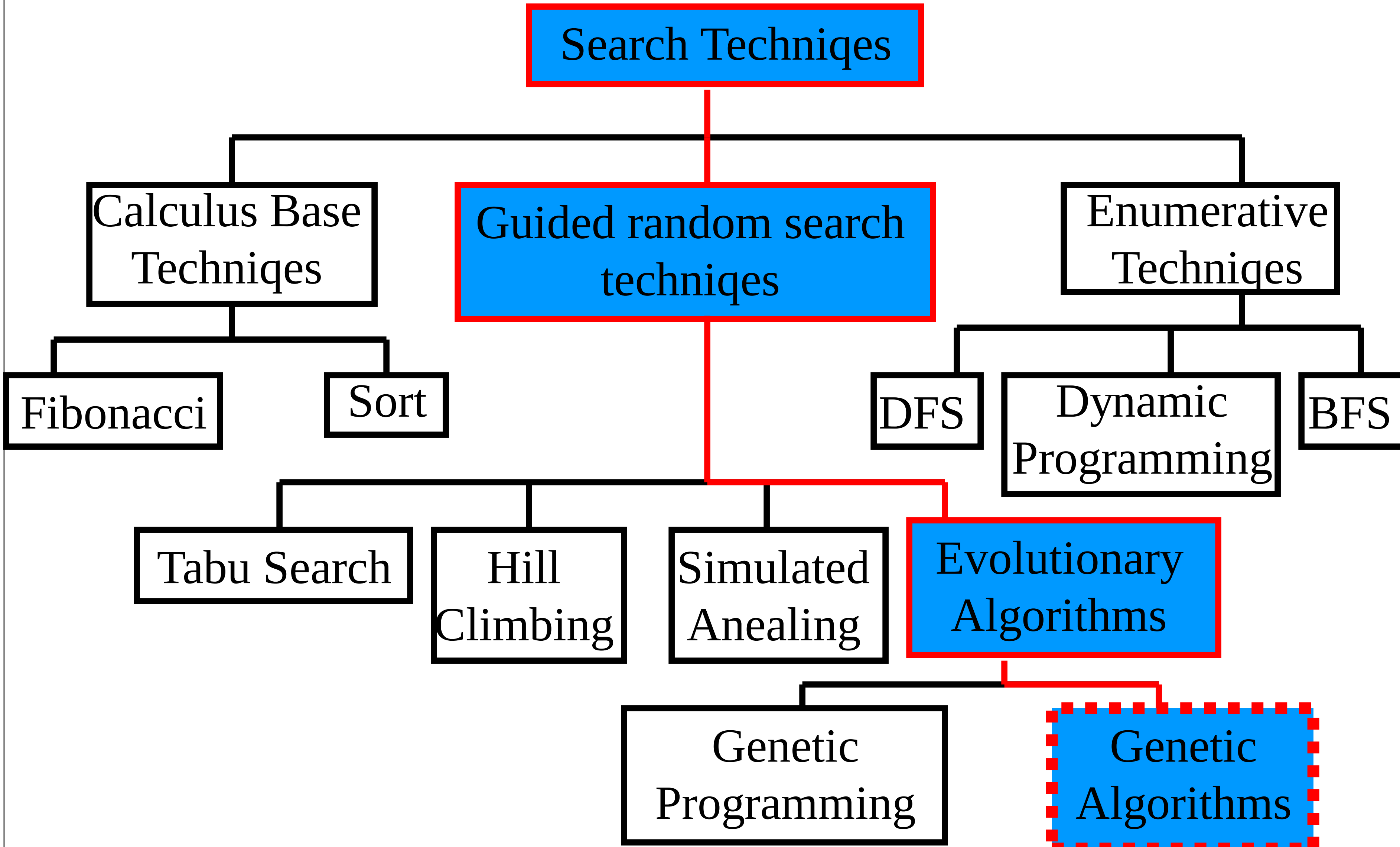
Genetic Algorithms (GA) OVERVIEW

- 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
- ***CrossOver*** decomposes two distinct solutions and then randomly mixes their parts to form novel solutions
- ***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, cross over 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

The Metaphor (cont)

The computer model introduces simplifications (relative to the real biological mechanisms),

BUT

surprisingly complex and interesting structures have emerged out of evolutionary algorithms

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

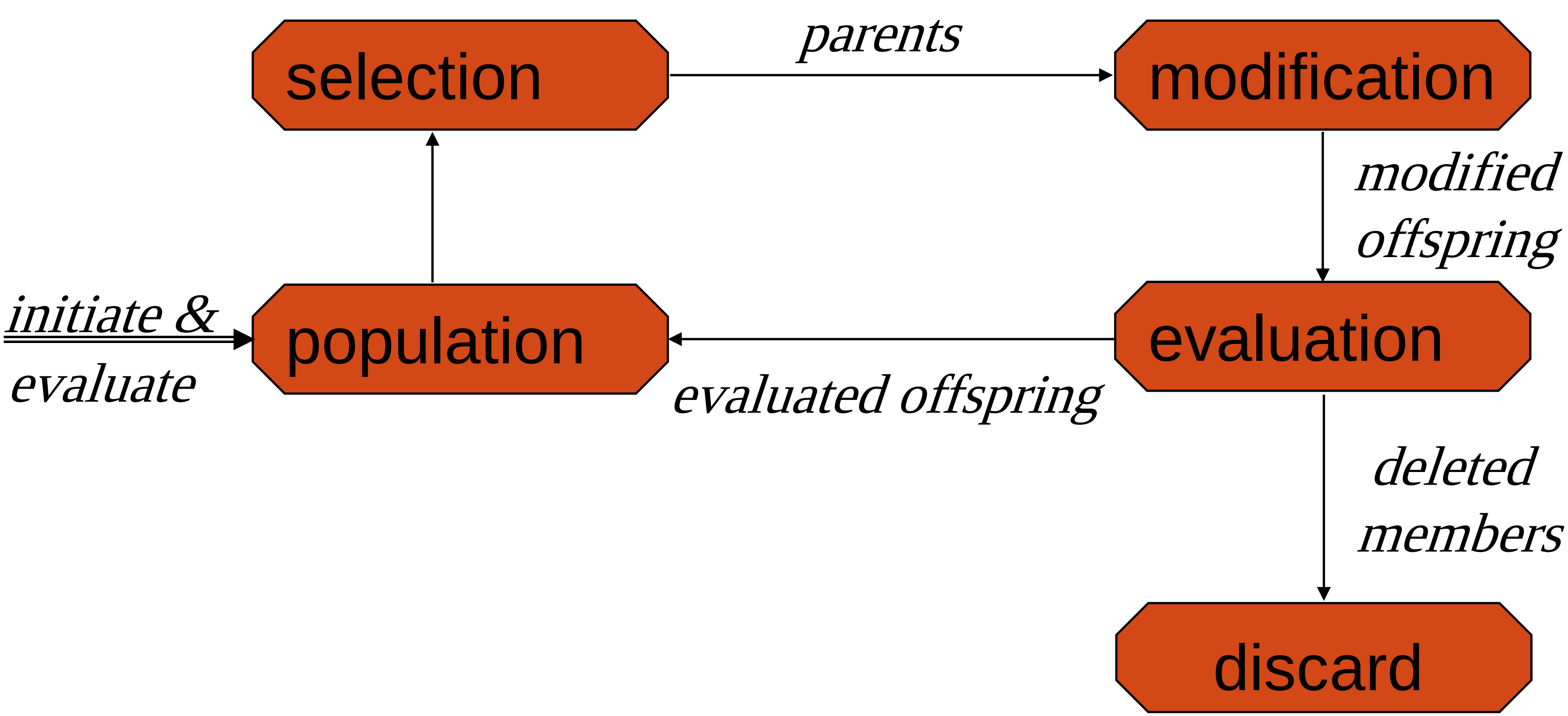
    mutate individuals

    evaluate the fitness of the modified individuals

    generate a new population

End while

The Evolutionary Cycle



Example:
the MAXONE problem

Suppose we want to maximize the number of ones in a string of l binary digits

Is it a trivial problem?

It may seem so because we know the answer in advance

However, we can think of it as maximizing the number of correct answers, each encoded by 1, to l yes/no difficult questions`

Example (cont)

- An individual is encoded (naturally) as a string of l binary digits
- The fitness f of a candidate solution to the MAXONE problem is the number of ones in its genetic code
- We start with a population of n random strings. Suppose that $l = 10$ and $n = 6$

Example (initialization)

We toss a fair coin 60 times and get the following initial population:

$$s_1 = 1111010101 \qquad f(s_1) = 7$$

$$s_2 = 0111000101 \qquad f(s_2) = 5$$

$$s_3 = 1110110101 \qquad f(s_3) = 7$$

$$s_4 = 0100010011 \qquad f(s_4) = 4$$

$$s_5 = 1110111101 \qquad f(s_5) = 8$$

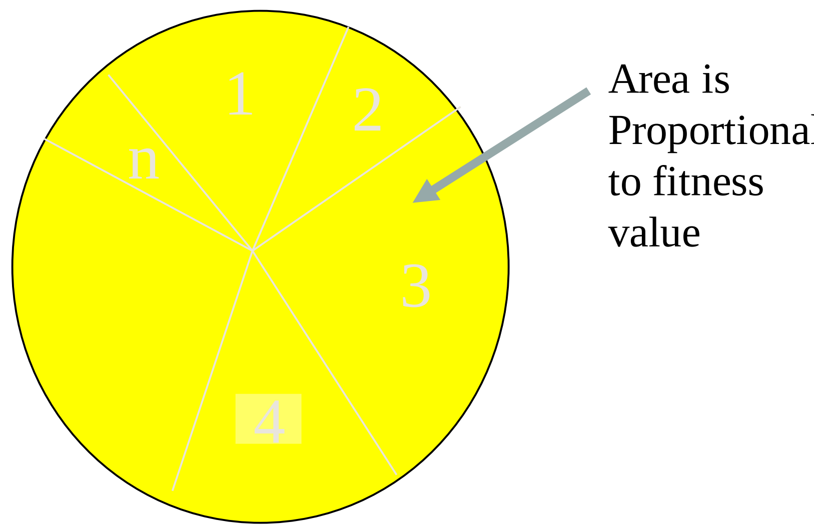
$$s_6 = 0100110000 \qquad f(s_6) = 3$$

Example (selection1)

Next we apply fitness proportionate selection with the roulette wheel method:

Individual i will have a $\frac{f(i)}{\sum_i f(i)}$ probability to be chosen

We repeat the extraction as many times as the number of individuals we need to have the same parent population size (6 in our case)



Example (selection2)

Suppose that, after performing selection, we get the following population:

$$s_1` = 1111010101 \qquad (s_1)$$

$$s_2` = 1110110101 \qquad (s_3)$$

$$s_3` = 1110111101 \qquad (s_5)$$

$$s_4` = 0111000101 \qquad (s_2)$$

$$s_5` = 0100010011 \qquad (s_4)$$

$$s_6` = 1110111101 \qquad (s_5)$$

Example (crossover1)

Next we mate strings for crossover. For each couple we decide according to crossover probability (for instance 0.6) whether to actually perform crossover or not

Suppose that we decide to actually perform crossover only for couples (s_1\`, $s_2$\`) and (s_5\`, $s_6$\`). For each couple, we randomly extract a crossover point, for instance 2 for the first and 5 for the second

Example (crossover2)

Before crossover:

s_1\` = 1111010101

s_2\` = 1110110101

s_5\` = 0100010011

s_6\` = 1110111101

After crossover:

s_1\`\` = 1110110101

s_2\`\` = 1111010101

s_5\`\` = 0100011101

s_6\`\` = 1110110011

Example (mutation1)

The final step is to apply random mutation: for each bit that we are to copy to the new population we allow a small probability of error (for instance 0.1)

Before applying mutation:

$$s_1`` = 1110110101$$

$$s_2`` = 1111010101$$

$$s_3`` = 1110111101$$

$$s_4`` = 0111000101$$

$$s_5`` = 0100011101$$

$$s_6`` = 1110110011$$

Example (mutation2)

After applying mutation:

$$s_1``` = 1110100101 \qquad f(s_1```) = 6$$

$$s_2``` = 1111110100 \qquad f(s_2```) = 7$$

$$s_3``` = 1110101111 \qquad f(s_3```) = 8$$

$$s_4``` = 0111000101 \qquad f(s_4```) = 5$$

$$s_5``` = 0100011101 \qquad f(s_5```) = 5$$

$$s_6``` = 1110110001 \qquad f(s_6```) = 6$$

Example (end)

In one generation, the total population fitness changed from 34 to 37, thus improved by ~9%

At this point, we go through the same process all over again, until a stopping criterion is met