Lecture 10

Instructor: Amit Kumar Das

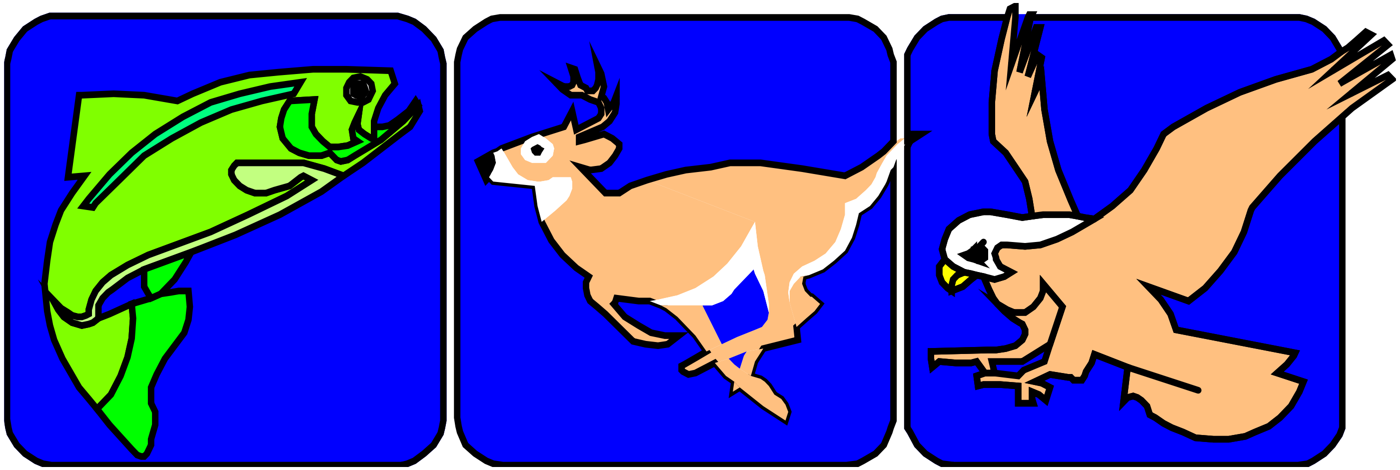
Senior Lecturer,

Department of Computer Science & Engineering,

East West University

Dhaka, Bangladesh.

A Simple Truth



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

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

Genetic Algorithm

- To understand the adaptive processes of natural systems
- To design artificial systems software that retains the robustness of natural systems
- Provide efficient, effective techniques for optimization and machine learning applications

Main Idea

- Take a population of candidate solutions to a given problem.
- Use operators inspired by the mechanisms of natural genetic variation.
- Apply selective pressure toward certain properties
- Evolve a more fit solution

GA Terminology

- Abstractions imported from biology
  - Chromosomes, Genes, Alleles
  - Fitness, Selection
  - Crossover, Mutation

GA Terminology

- In the spirit – but not the letter – of biology
  - GA chromosomes are strings of genes
    - Each gene has a number of alleles; i.e., settings
  - Each chromosome is an encoding of a solution to a problem
  - A population of such chromosomes is operated on by a GA

Encoding

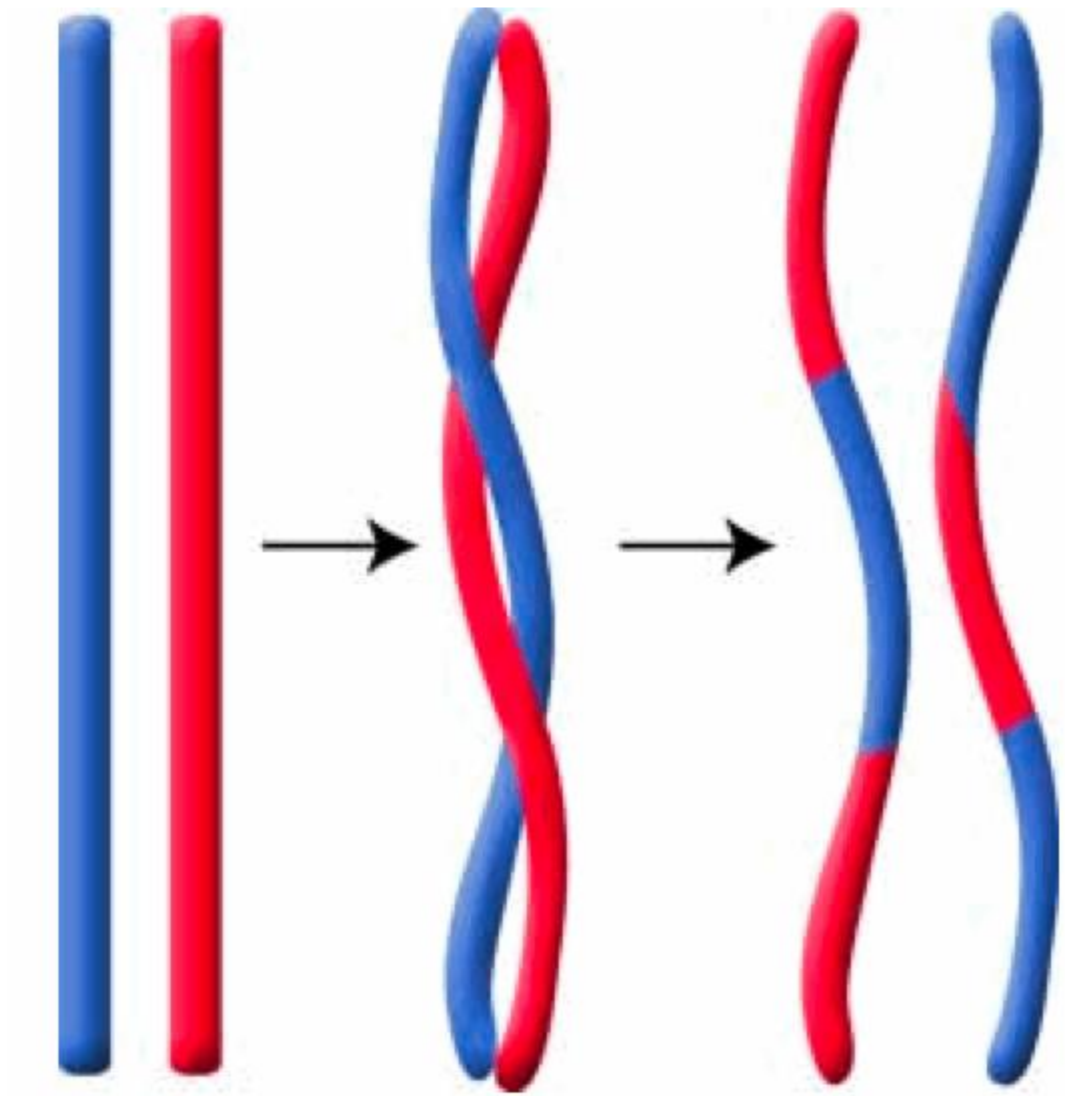
- A data structure for representing candidate solutions
  - Often takes the form of a bit string
  - Usually has internal structure; i.e., different parts of the string represent different aspects of the solution.

Crossover

- Mimics biological recombination
  - Some portion of genetic material is swapped between chromosomes
  - Typically the swapping produces an offspring

- Mechanism for the dissemination of "building blocks" (schemas)

Crossover

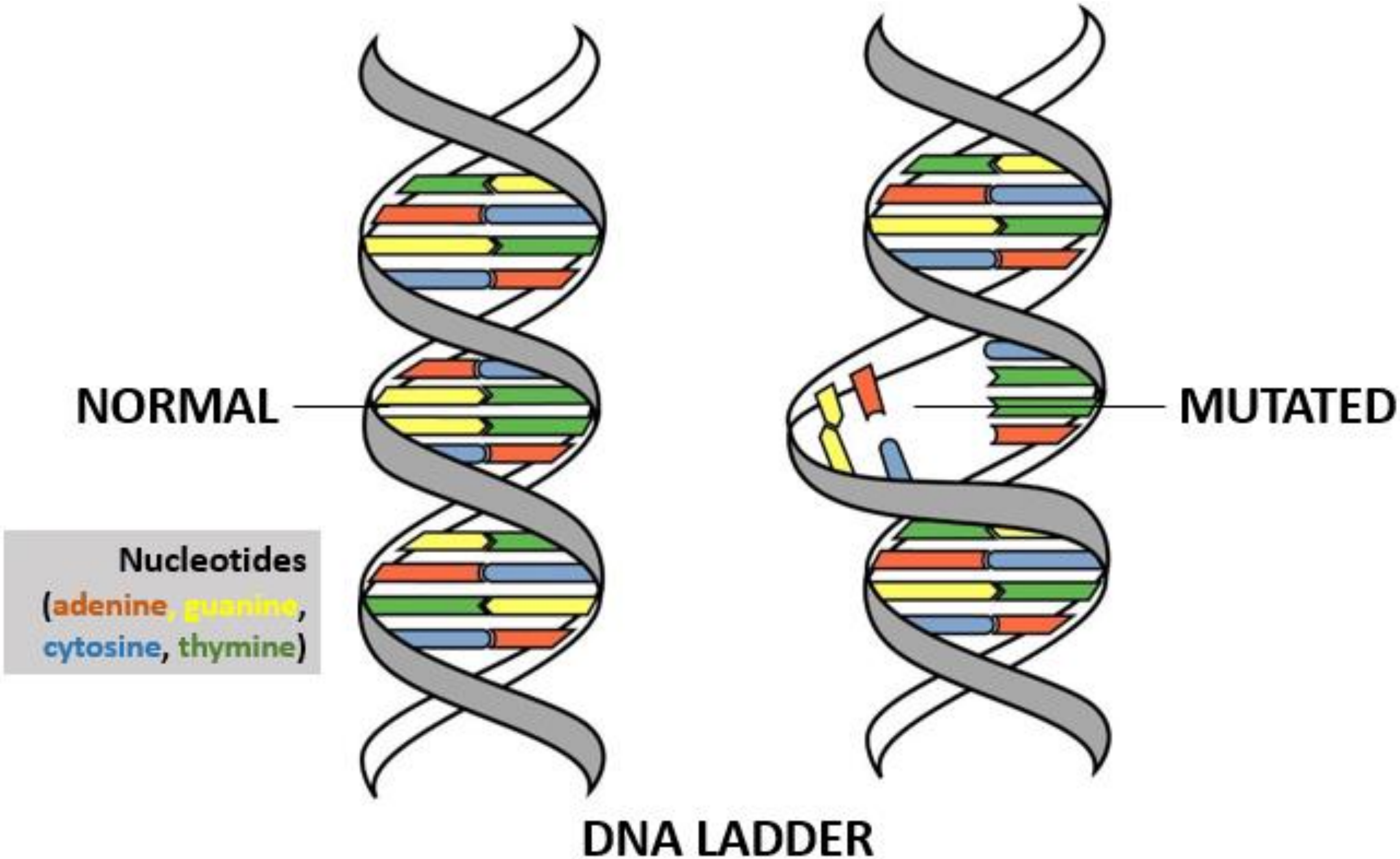


Mutation

- Selects a random locus – gene location – with some probability and alters the allele at that locus

- The intuitive mechanism for the preservation of variety in the population

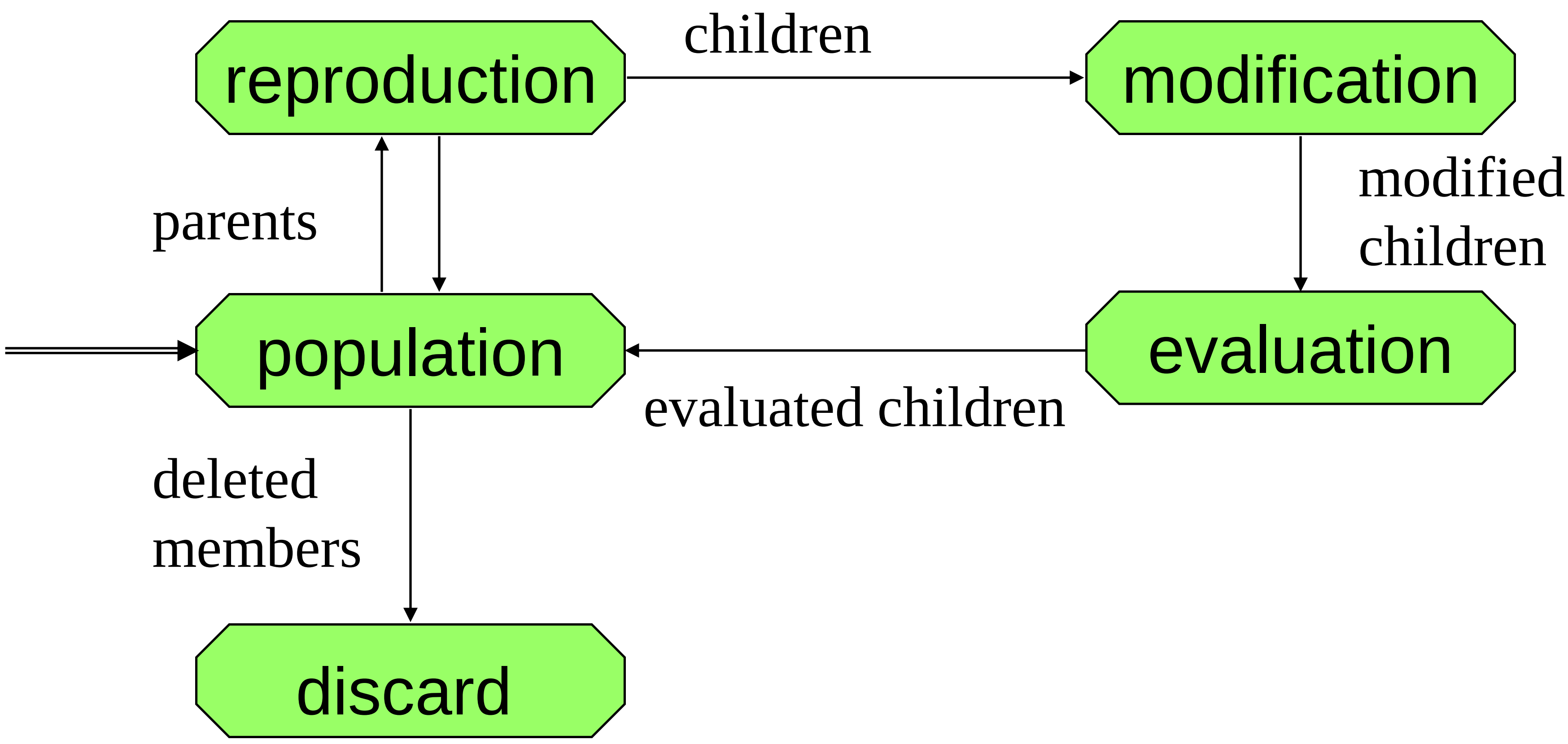
Mutation


NORMAL
MUTATED
Nucleotides
(adenine, guanine,
cytosine, thymine)
DNA LADDER

Fitness

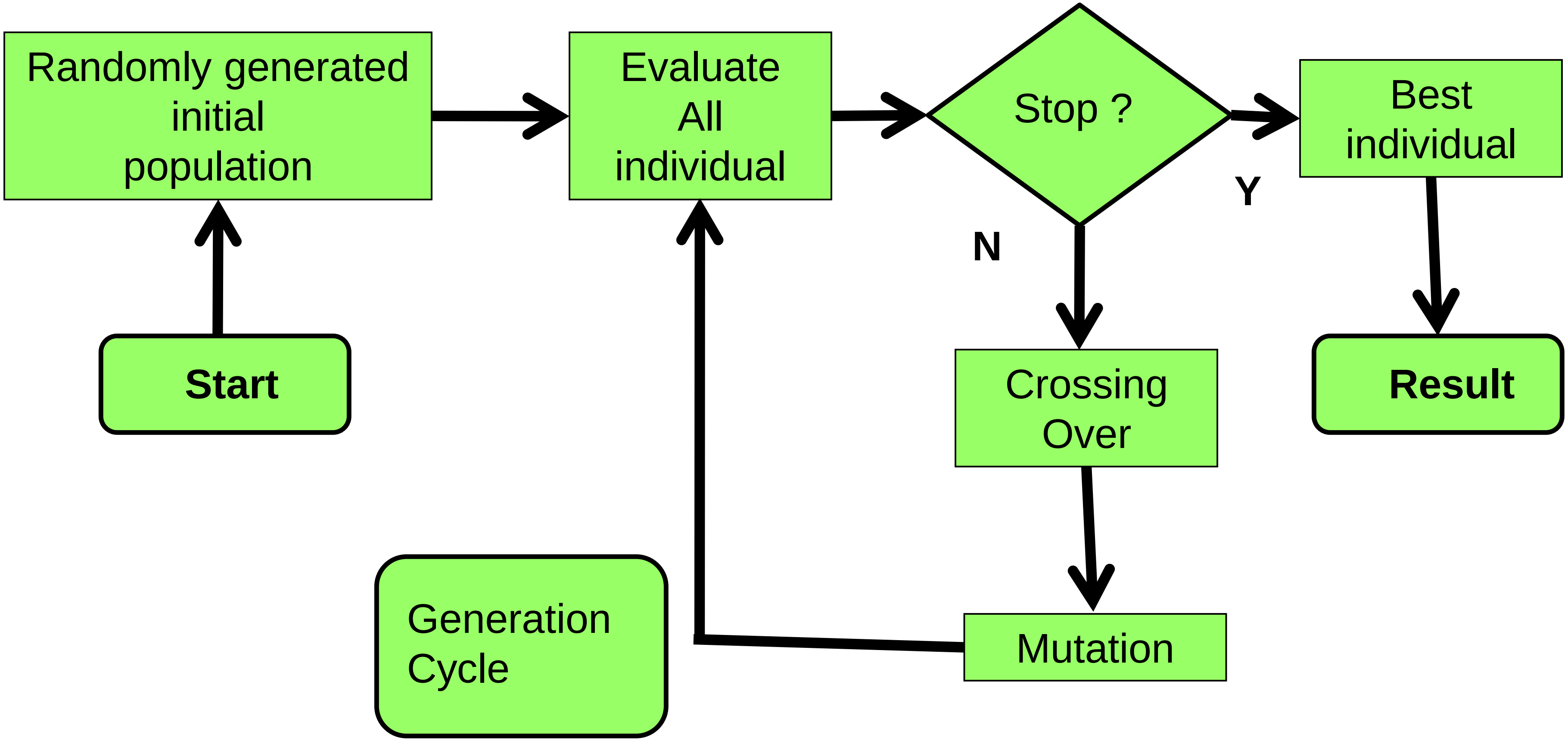
- A measure of the goodness of the organism
- Expressed as the probability that the organism will live another cycle (generation)
- Basis for the natural selection simulation
  - Organisms are selected to mate with probabilities proportional to their fitness
- Probabilistically better solutions have a better chance of conferring their building blocks to the next generation (cycle)

The GA cycle



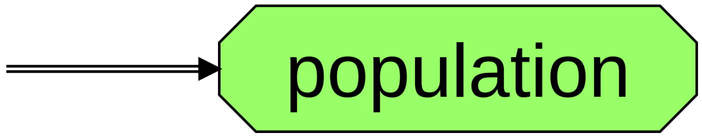
reproduction
children
modification
parents
modified
children
population
evaluation
evaluated children
deleted
members
discard

The GA cycle



Randomly generated initial population
Evaluate All individual
Stop ?
Best individual
Y
N
Start
Crossing Over
Result
Generation Cycle
Mutation

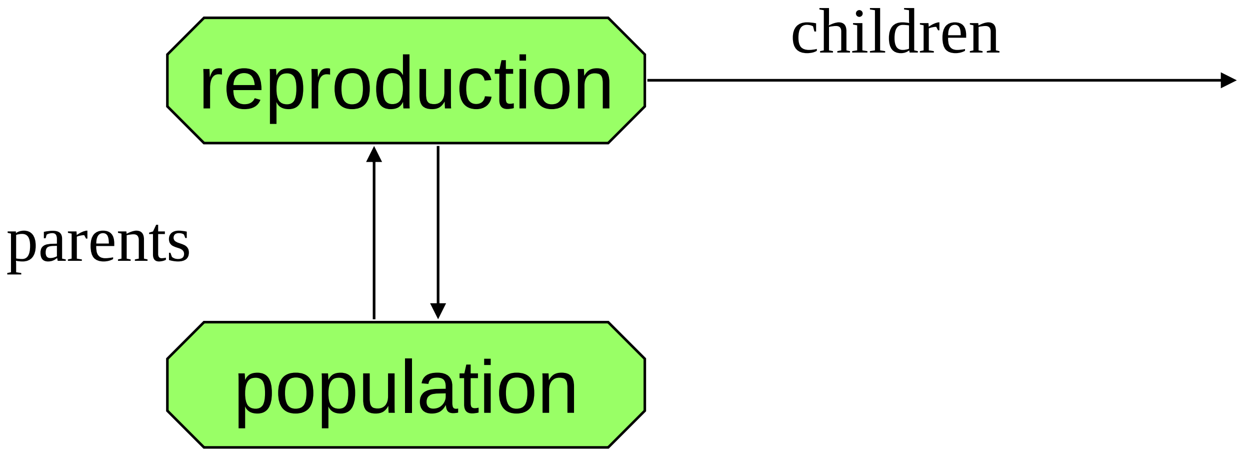
Population



Chromosomes could be:

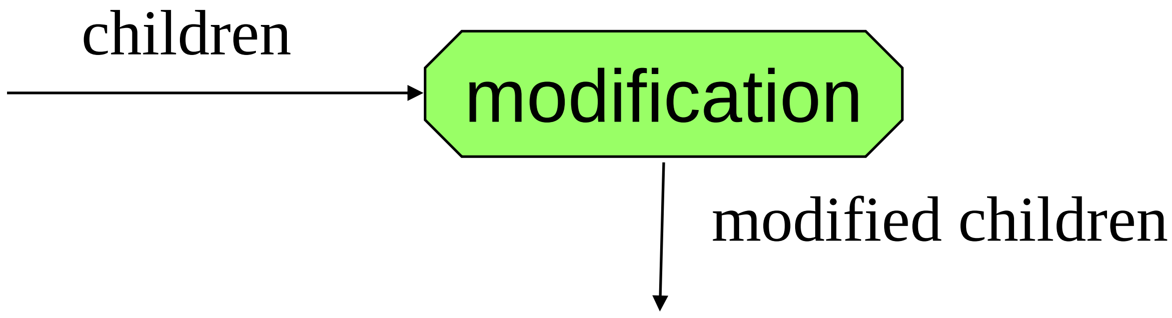
- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Reproduction



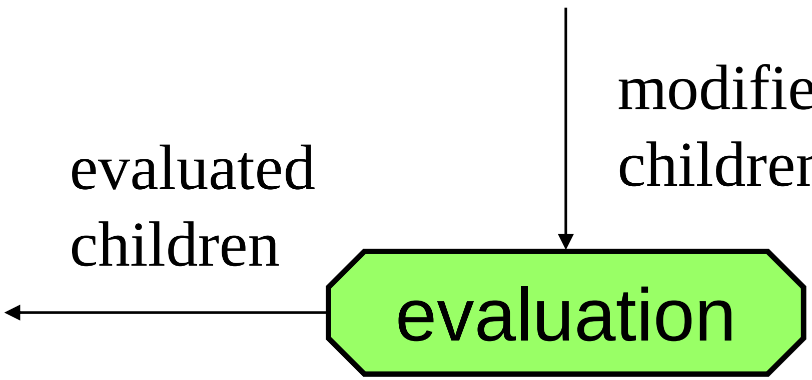
Parents are selected at random with selection chances biased in relation to chromosome evaluations.

Modification



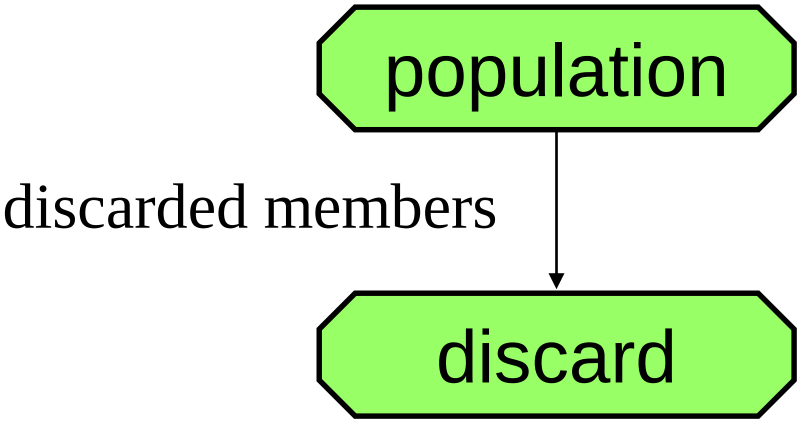
- Modifications are stochastically triggered
- Operator types are:
  – Mutation
  – Crossover (recombination)

Evaluation



- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

Deletion



- *Generational* GA:
  entire populations replaced with each iteration
- *Steady-state* GA:
  a few members replaced at each generation

The good genes (features) of the parents are passed onto the children

Fitness Function: Pairs of nonattacking queens

That way, higher scores are better.

23


+
=
3
2
1
1 2 3 . . .
23
24748552

Represent states and compute fitness function.

24748552	24
	+
32752411	23
	+
24415124	20
	+
32543213	11
	77

(a)

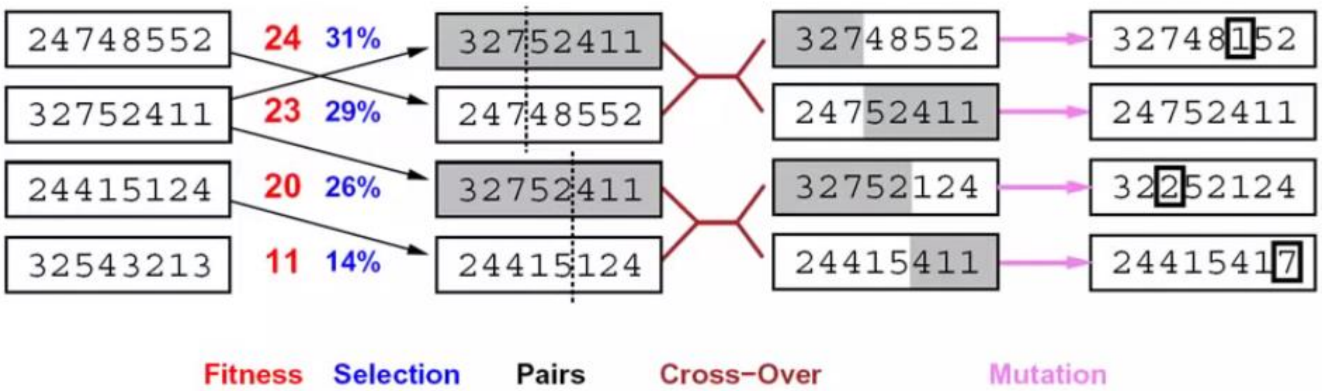
Initial Population

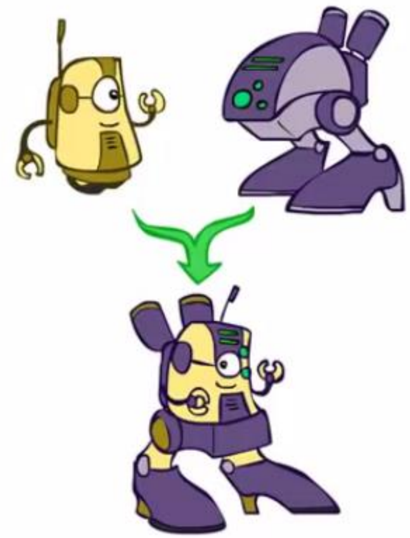
Compute probability of being chosen (from fitness function).

24748552	24	31%
32752411	23	29%
24415124	20	26%
32543213	11	14%

(a)
Initial Population

Genetic Algorithms





- Genetic algorithms use a natural selection metaphor
  - Keep best N hypotheses at each step (selection) based on a fitness function
  - Also have pairwise crossover operators, with optional mutation to give variety
- Possibly the most misunderstood, misapplied (and even maligned) technique around

Example:
the MAXONE problem

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

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	=	1	1	1	1	0	1	0	1	0	1	$f(S_1)$	= 7
S_2	=	0	1	1	1	0	0	0	1	0	1	$f(S_2)$	= 5
S_3	=	1	1	1	0	1	1	0	1	0	1	$f(S_3)$	= 7
S_4	=	0	1	0	0	0	1	0	0	1	1	$f(S_4)$	= 4
S_5	=	1	1	1	0	1	1	1	1	0	1	$f(S_5)$	= 8
S_6	=	0	1	0	0	1	1	0	0	0	0	$f(S_6)$	= 3
Fitness													= 34

Example (selection)

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

S_1'	=	1	1	1	1	0	1	0	1	0	1	(S_1)	=	7
S_2'	=	1	1	1	0	1	1	0	1	0	1	(S_3)	=	7
S_3'	=	1	1	1	0	1	1	1	1	0	1	(S_5)	=	8
S_4'	=	0	1	1	1	0	0	0	1	0	1	(S_2)	=	5
S_5'	=	0	1	0	0	0	1	0	0	1	1	(S_4)	=	4
S_6'	=	0	1	0	0	1	1	0	0	0	0	(S_6)	=	3
Fitness													=	34

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'	=	1	1	1	1	0	1	0	1	0	1
S_2'	=	1	1	1	0	1	1	0	1	0	1

S_5'	=	0	1	0	0	0	1	0	0	1	1
S_6'	=	0	1	0	0	1	1	0	0	0	0

After crossover

S_1''	=	1	1	1	0	1	1	0	1	0	1
S_2''	=	1	1	1	1	0	1	0	1	0	1

S_5''	=	0	1	0	0	0	1	0	0	0	0
S_6''	=	0	1	0	0	1	1	0	0	1	1

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''	=	1	1	1	0	1	1	0	1	0	1
S_2''	=	1	1	1	1	0	1	0	1	0	1
S_3''	=	1	1	1	0	1	1	1	1	0	1
S_4''	=	0	1	1	1	0	0	0	1	0	1
S_5''	=	0	1	0	0	0	1	0	0	0	0
S_6''	=	0	1	0	0	1	1	0	0	1	1

Example (mutation2)

After applying mutation:

S_1'''	=	1	1	1	0	1	1	1	1	0	1	$f(S_1''')$	=	8
S_2'''	=	1	1	1	1	1	1	0	1	0	0	$f(S_2''')$	=	7
S_3'''	=	1	1	1	0	1	1	1	1	1	1	$f(S_3''')$	=	9
S_4'''	=	0	1	1	1	0	0	0	1	0	1	$f(S_4''')$	=	5
S_5'''	=	0	1	0	0	0	1	0	0	0	1	$f(S_5''')$	=	3
S_6'''	=	0	1	0	0	1	1	0	1	1	1	$F(S_6''')$	=	5
Fitness													=	37

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

Thank You