Genetic Algorithms

-- Lecture based on Martin Pelikan's MEDAL technical report 2010007 available at the tutoria --

Fernando Lobo

University of Algarve

Outline

- Components of a GA
- Simulation of a GA by hand and by computer.

Components of a GA

- Initialization
  - usually at random, but can use prior knowledge.
- Selection
  - give preference to better solutions.
- Variation
  - create new solutions through crossover and mutation.
- Replacement
  - combine original population with newly created solutions.

Selection

- Various methods can be used.
- Main idea is to make copies of solutions that perform better at the expense of solutions that perform worse.
- Two major classes of methods:
  - Proportionate
  - Ordinal

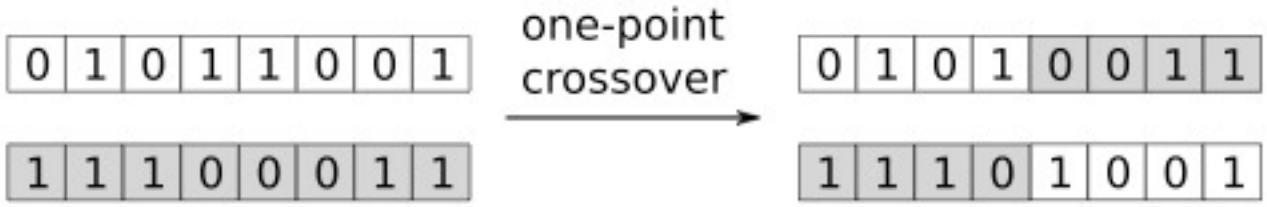
Variation

- Two major operators:
  - Crossover
  - Mutation

Crossover

- Combines pieces of promising solutions by exchanging some of their parts.
- Typically applied to pairs of solutions with a specified probability *Pc*
- *Pc* is a parameter of the GA
  - typically in the range [0.6, 0.95]

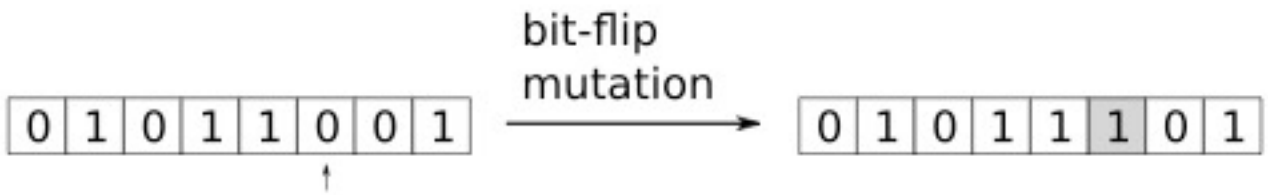
Example: 1-pt crossover



Mutation

- Makes a slight perturbation of a solution.
- A commonly used operator for binary coded representations is *bit-flip* mutation.
  - each bit is independently flipped with probability *Pm*
- *Pm* is another GA parameter
  - typically a small value in order to change only one or a few bits (on average) per solution.

Example: bit-flip mutation



Replacement

- Newly created solutions replace some (or all) of the current solutions.

Pseudocode

```
Genetic algorithm {
    t = 0;
    generate initial population P(0);
    while (not done) {
        select population of promising solutions S(t) from P(t);
        apply variation operators on S(t) to create O(t);
        create P(t+1) by combining O(t) and P(t);
        t = t+1;
    }
}
```

Common GA terminology

Term	Alternative name(s)
candidate solution	individual, chromosome, string
decision variable	variable, locus, string position
value of decision variable	bit, allele
measure of solution quality	fitness function, objective function
numeric representation of solution quality	fitness, fitness value
population of promising solutions, $S(t)$	parent population, parents, selected solutions
population of new solutions, $O(t)$	offspring population, offspring, children
iteration	generation

Simulating a GA by hand (1)

- Let's look at a simple problem
- Onemax problem, fitness is the number of 1s in a binary string.

$$onemax(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i,$$

- It is a unimodal problem, very easy to solve (we don't even need a GA!)

Simulating a GA by hand (2)

- We will use binary tournament selection, one-point crossover, bit-flip mutation, and full replacement.

- Parameters

parameter	description	value
n	string length	8
N	population size	10
p_c	crossover probability	0.6
p_m	mutation probability	1/8

Simulating a GA by hand (3)

- Create N=10 binary strings of length n=8

	candidate solution	fitness		candidate solution	fitness
1	10111001	5	6	00001010	2
2	00010010	2	7	11011000	4
3	01101001	4	8	11111000	5
4	00001001	2	9	11111000	5
5	00100110	3	10	00011001	3

Simulating a GA by hand (4)

- Do selection by randomly choosing N=10 pairs of individuals to compete with each other.

	tournament	winner	fitness		tournament	winner	fitness
1	11111000 & 00011001	11111000	5	6	00010010 & 11111000	11111000	5
2	00010010 & 00011001	00011001	3	7	01101001 & 10111001	10111001	5
3	11011000 & 10111001	10111001	5	8	00001001 & 00010010	00010010	2
4	01101001 & 00001010	01101001	4	9	00100110 & 11011000	11011000	4
5	00100110 & 11111000	11111000	5	10	10111001 & 00011001	10111001	5

Simulating a GA by hand (5)

- Take the winners and do crossover and mutation

	parents	crossing site	after crossover	after mutation	fitness
1	11111000	3	11011001	11111001	6
2	00011001		00111000	00111000	3
3	10111001	-	10111001	11111001	6
4	01101001		01101001	00111101	5
5	11111000	2	11111000	11110000	4
6	11111000		11111000	11111000	5
7	10111001	-	10111001	10011011	5
8	00010010		00010010	10010010	3
9	11011000	6	11011001	11010001	4
10	10111001		10111000	10111000	4

Simulating a GA by hand (6)

- Since we are simulating with full replacement, the resulting population (after crossover + mutation) replaces the original population.
- The average fitness of solutions improved from 3.5 to 4.5
- And we got some solutions (with fitness 6) better than any other solution we had before.

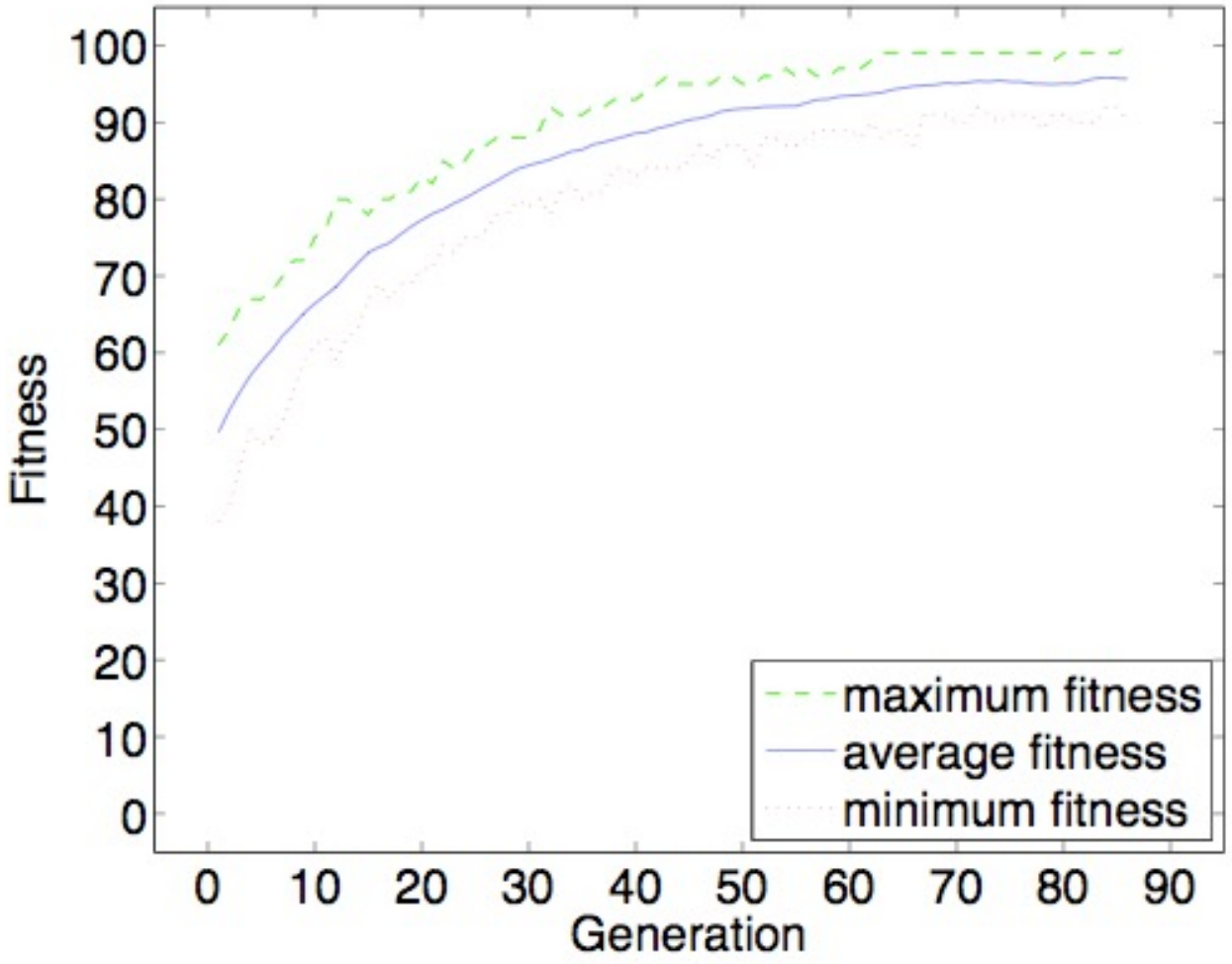
Simulating a GA by hand (7)

- Simulating more generations would improve the solutions further.
- Hopefully we would get the optimum: 11111111

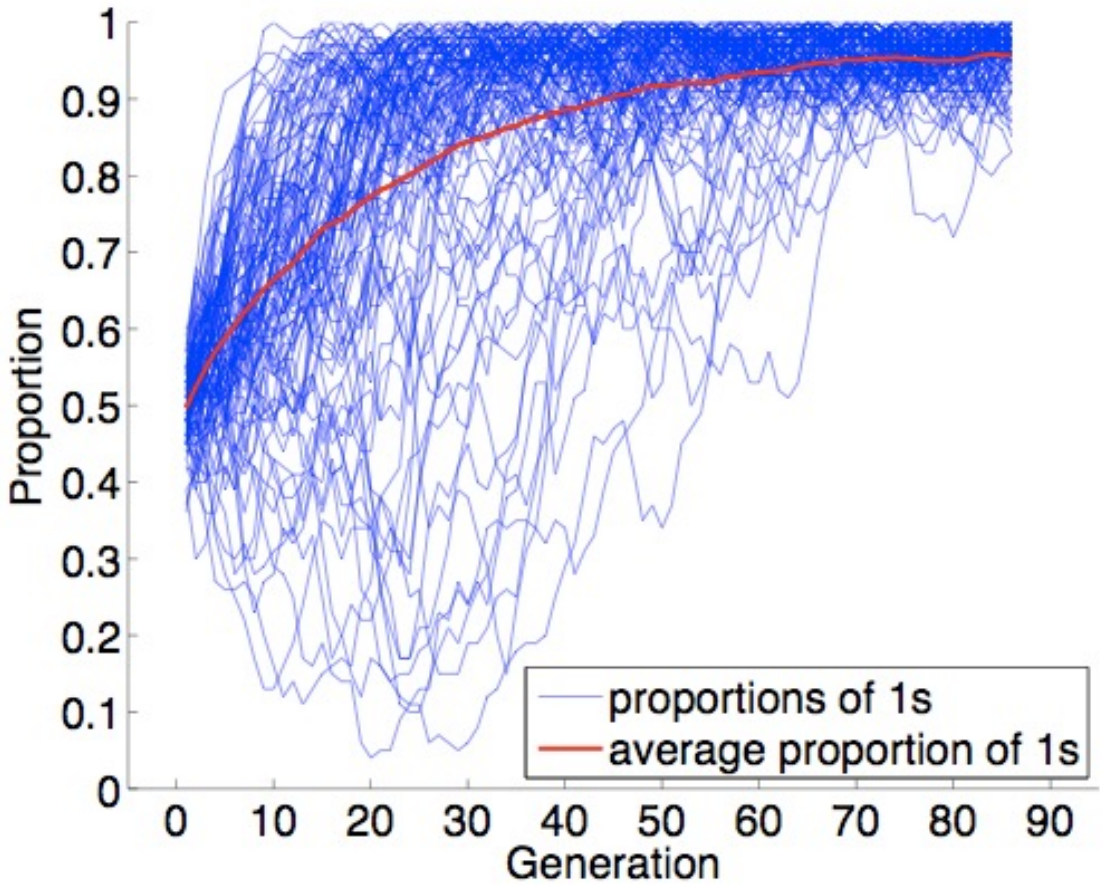
Simulating a larger onemax problem

- String length n=100
- Population size N=100
  - Larger problem → larger population
- Pc=0.6, Pm=1/100
- By hand? No way! Let's use a computer.

Fitness values through generations



Proportion of 1s for the various variables



So what?

- Onemax is a very easy problem.
- GA can solve it easily.
- Not so easily with other problems.
- Want to give it a try?
  - you can program a GA now.