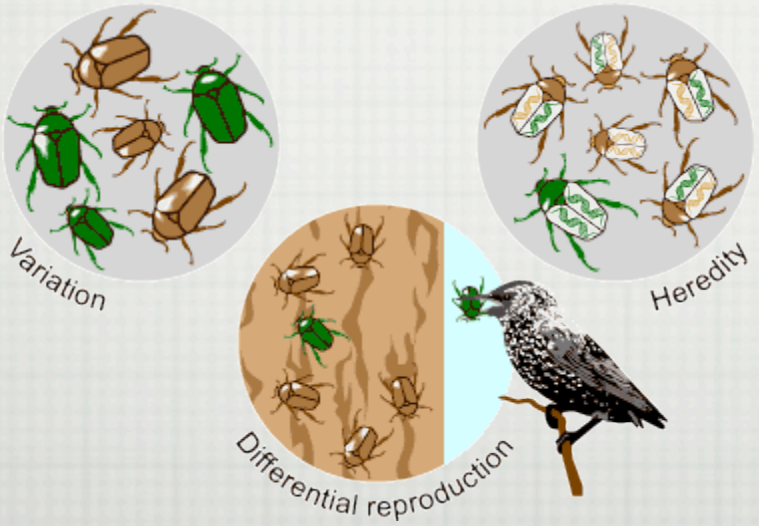
PHY-4810
COMPUTATIONAL PHYSICS

LECTURE 16: INTRODUCTION TO GENETIC ALGORITHMS

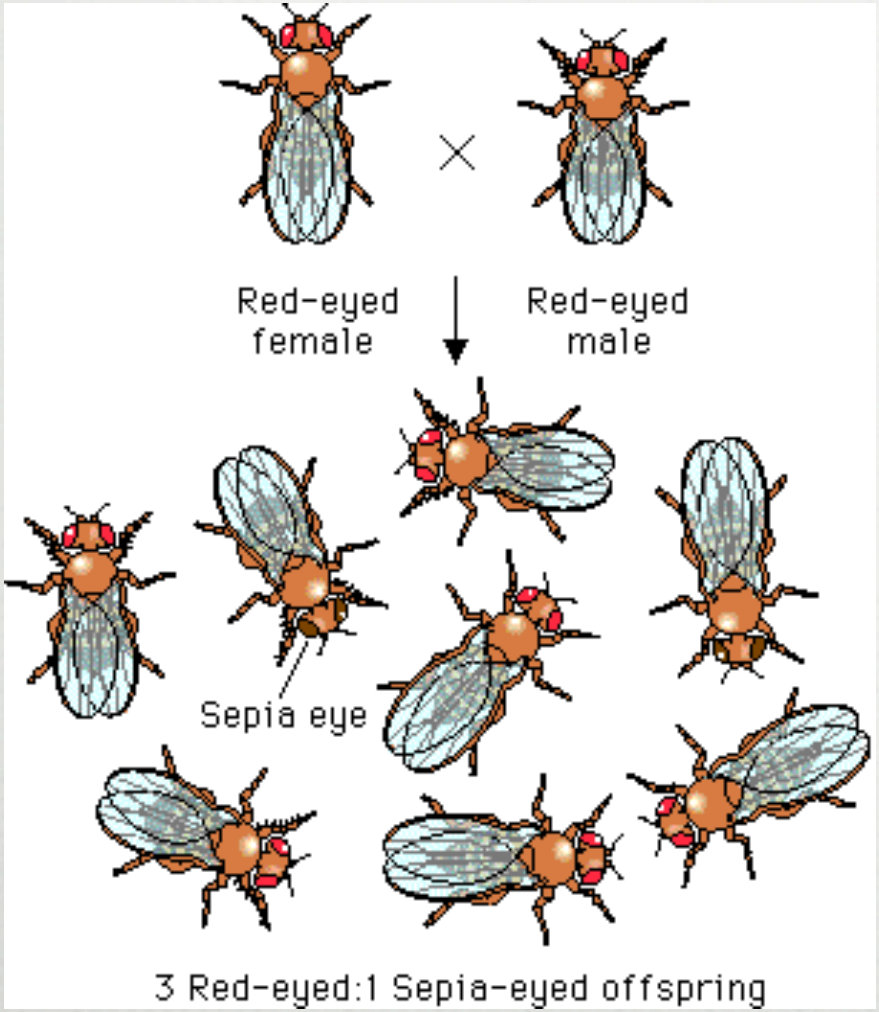
OVERSIMPLIFIED DESCRIPTION OF HOW EVOLUTION WORKS IN BIOLOGY

- Organisms produce a number of offsprings which are almost like themselves
- Variation may be due to random changes (mutations)
  - Variation may be due to sexual reproduction (offsprings have some characteristics from each parent)
  - Some of these offsprings may survive to produce offsprings of their own—some won't

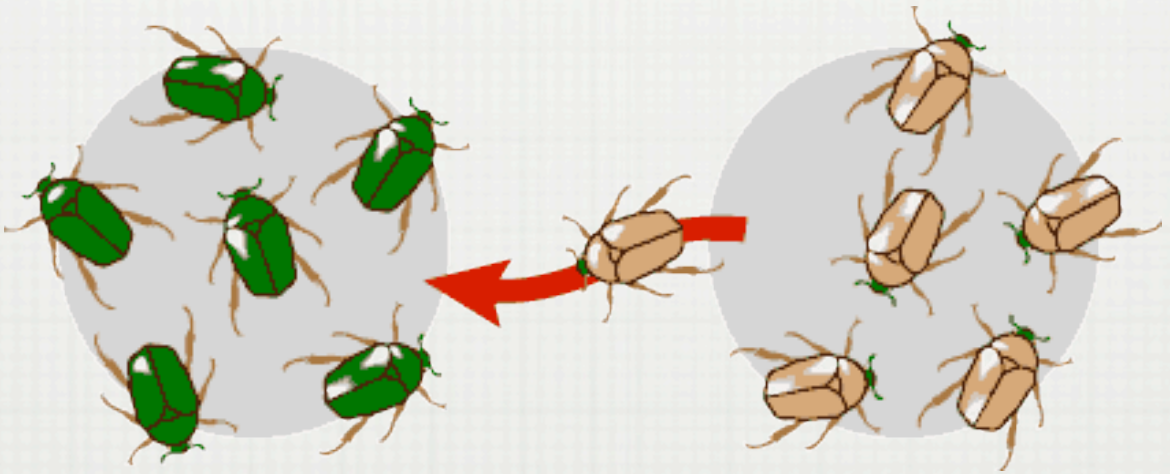


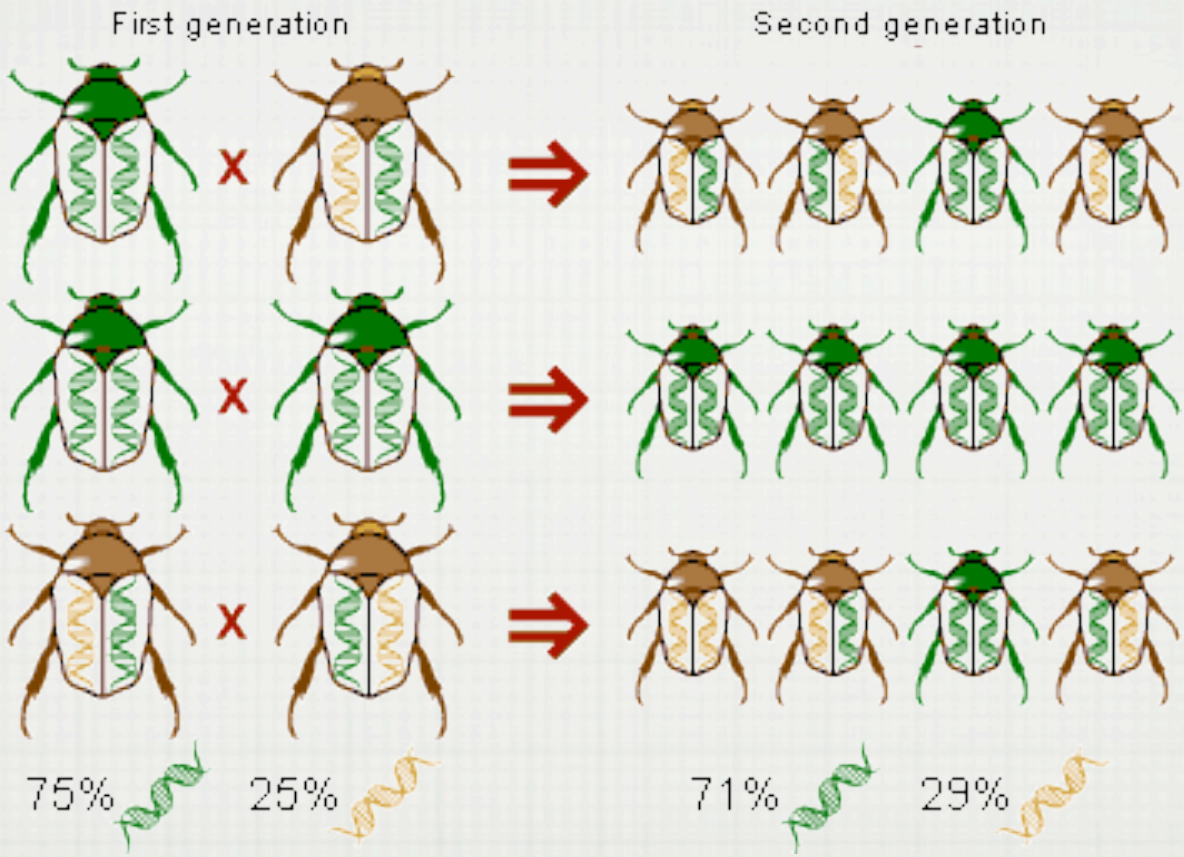
- The "better adapted" offspring are more likely to survive
- Over time, later generations become better and better adapted
- Genetic algorithms use this same process to "evolve" better programs

ORGANISMS PRODUCE OFFSPRINGS WHICH ARE ALMOST LIKE THEMSELVES



GENE FLOW, DRIFT (MIGRATION, CROSSOVER)

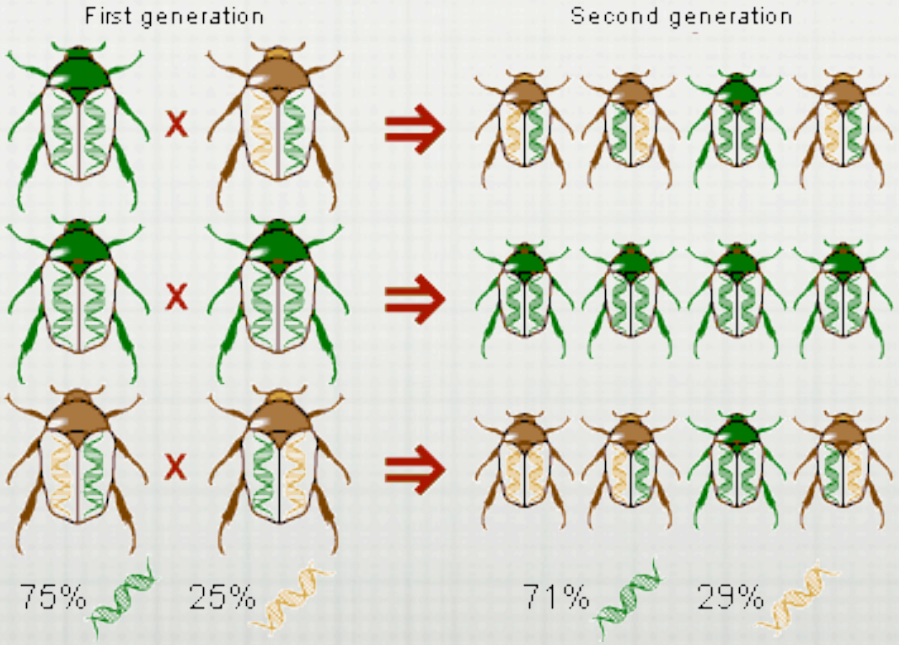




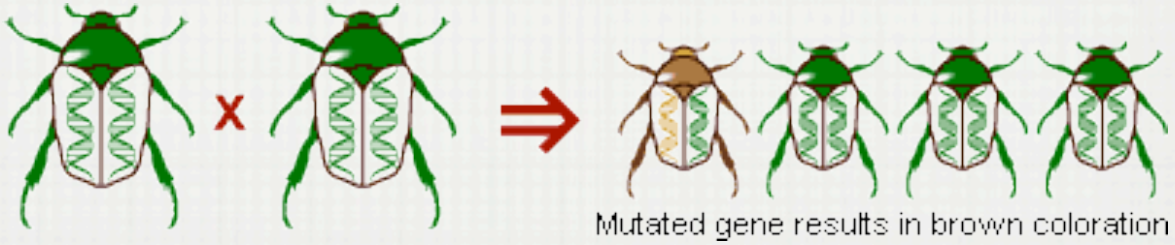
Offsprings and parents

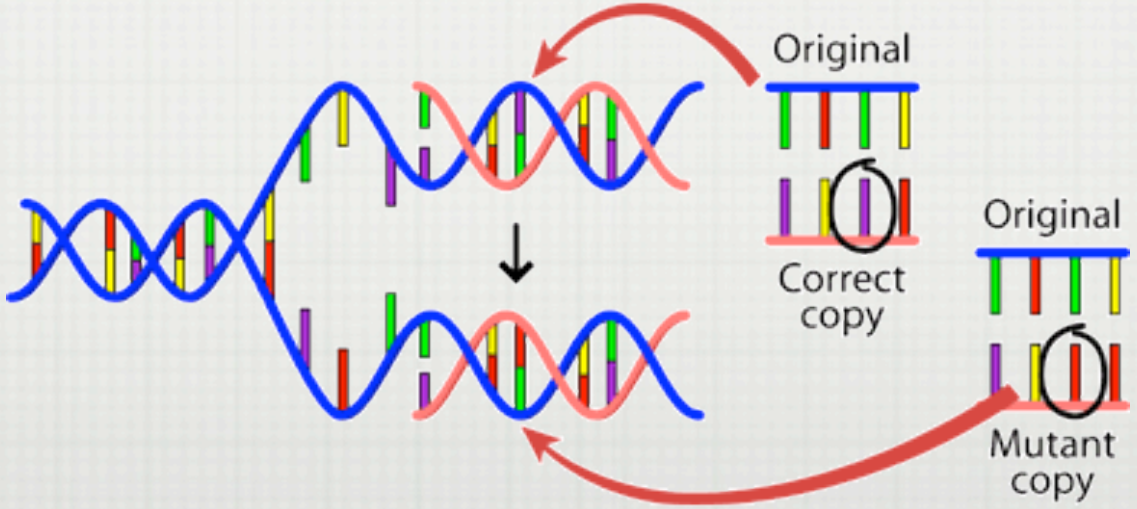
VARIATIONS DUE TO REPRODUCTION

- **Variations may be due to sexual reproduction (offsprings have some characteristics from each parent)**
- **Some of these offsprings may survive to produce offsprings of their own—some won't**



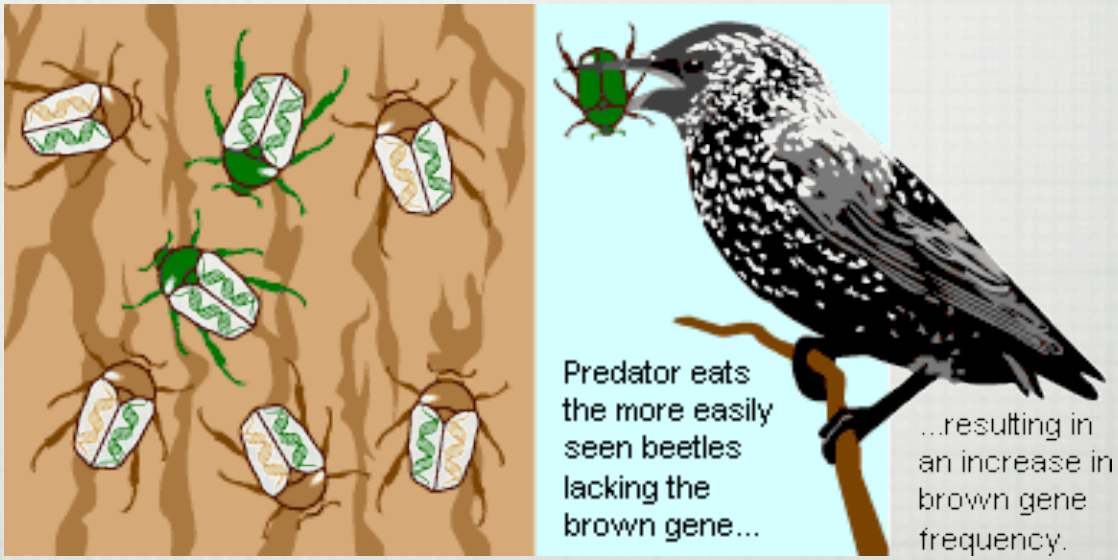
RANDOM VARIATIONS (MUTATIONS)

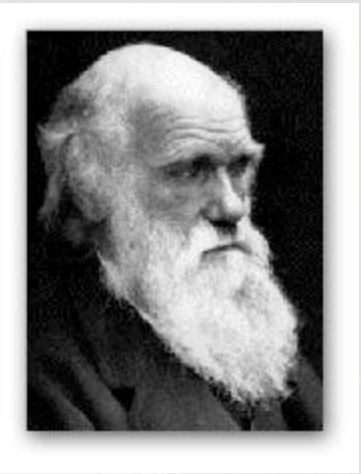




ADAPTATION: NATURAL SELECTION

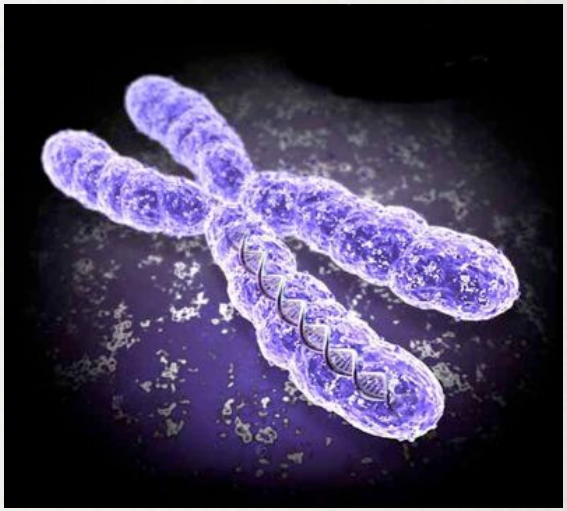
- **The "better adapted" offspring are more likely to survive "natural selection"**
- **Over time, later generations become better and better adapted**

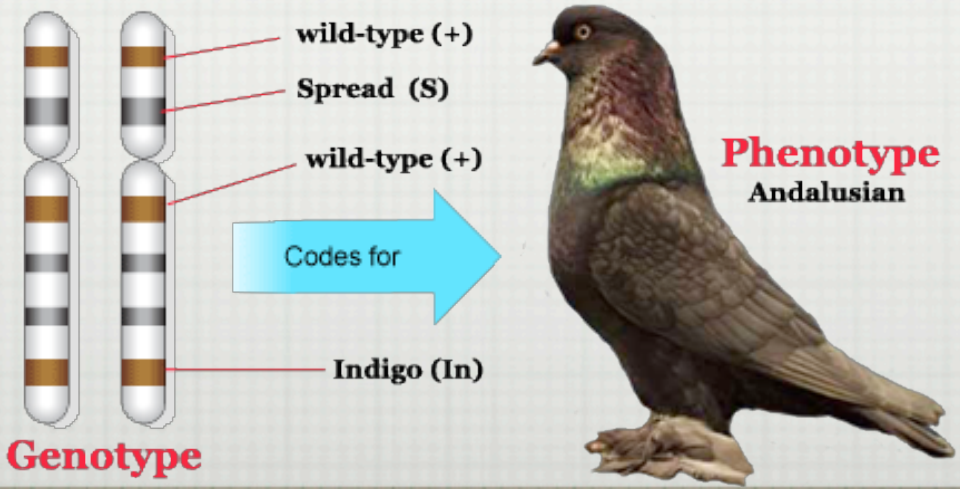




GENOTYPES AND PHENOTYPES

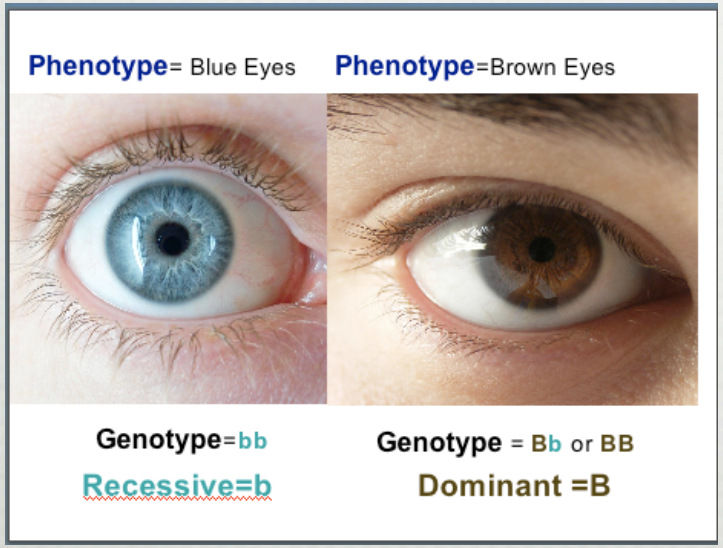
- Genes are the basic "instructions" for building an organism
  - A chromosome is a sequence of genes
  - Biologists distinguish between an organism's
    - **genotype** (the genes and chromosomes) and its
    - **phenotype** (what the organism actually is like)

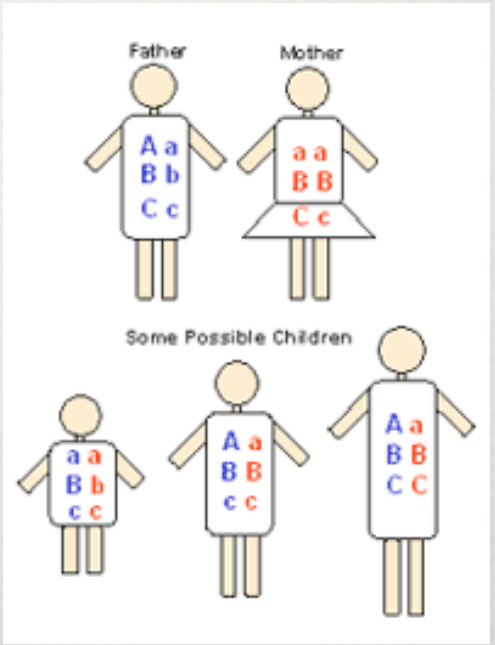




GENOTYPES AND PHENOTYPES: EXAMPLE

- **You might have genes to be tall, but never grow to be tall for other reasons (such as poor diet)**
- **Similarly, "genes" may describe a possible solution to a problem, without actually being the solution**





THE BASIC GENETIC ALGORITHM

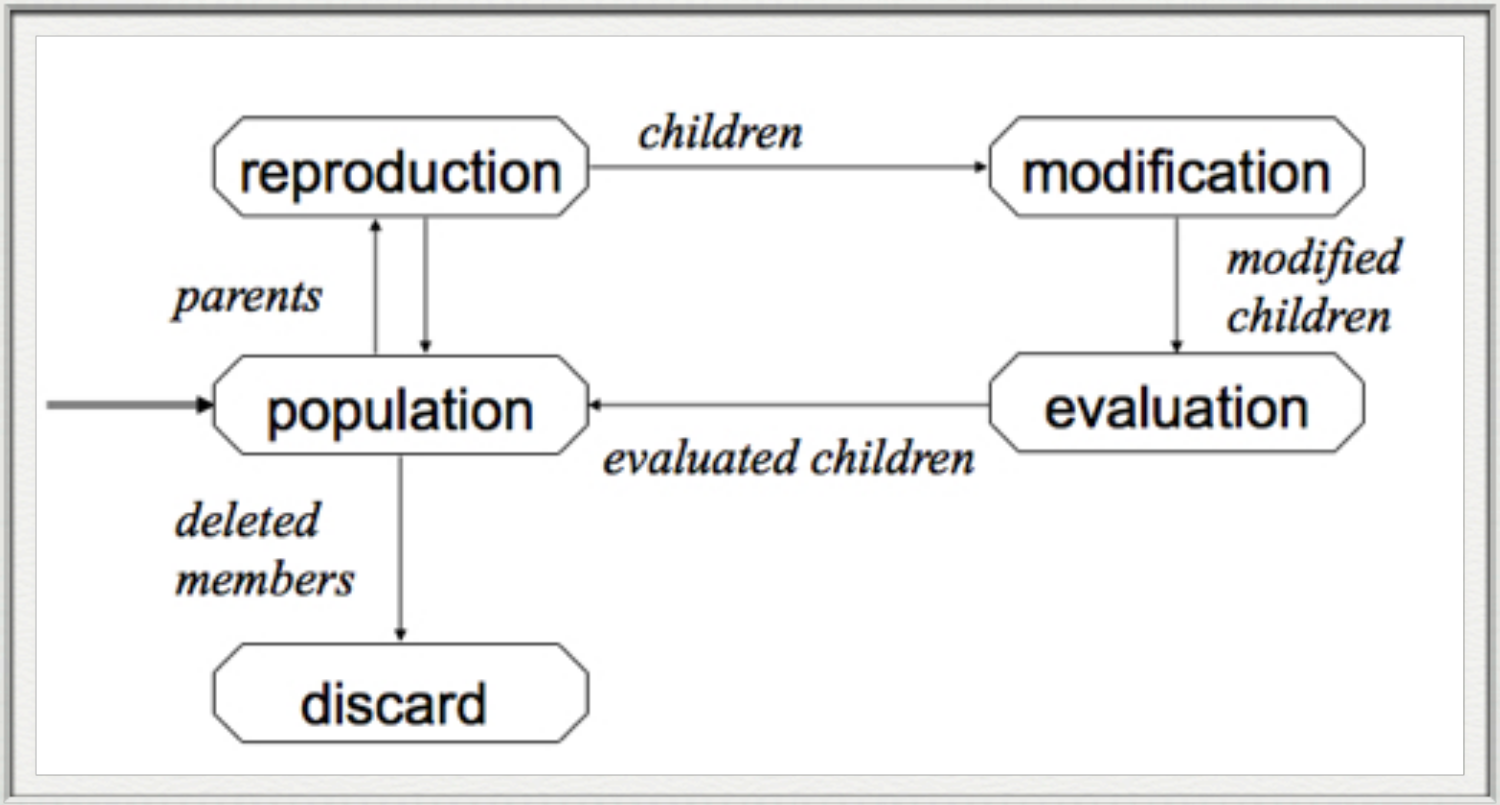
- Start with a large "population" of randomly generated "attempted solutions" to a problem
- Repeatedly do the following:
  1. Evaluate each of the attempted solutions
  2. Keep a subset of these solutions (the "best" ones)
  3. Use these solutions to generate a new population
  4. Mutate to add diversity to the population
- Quit when you have a satisfactory solution (or you run out of time)

COMPONENTS OF GENETIC ALGORITHM

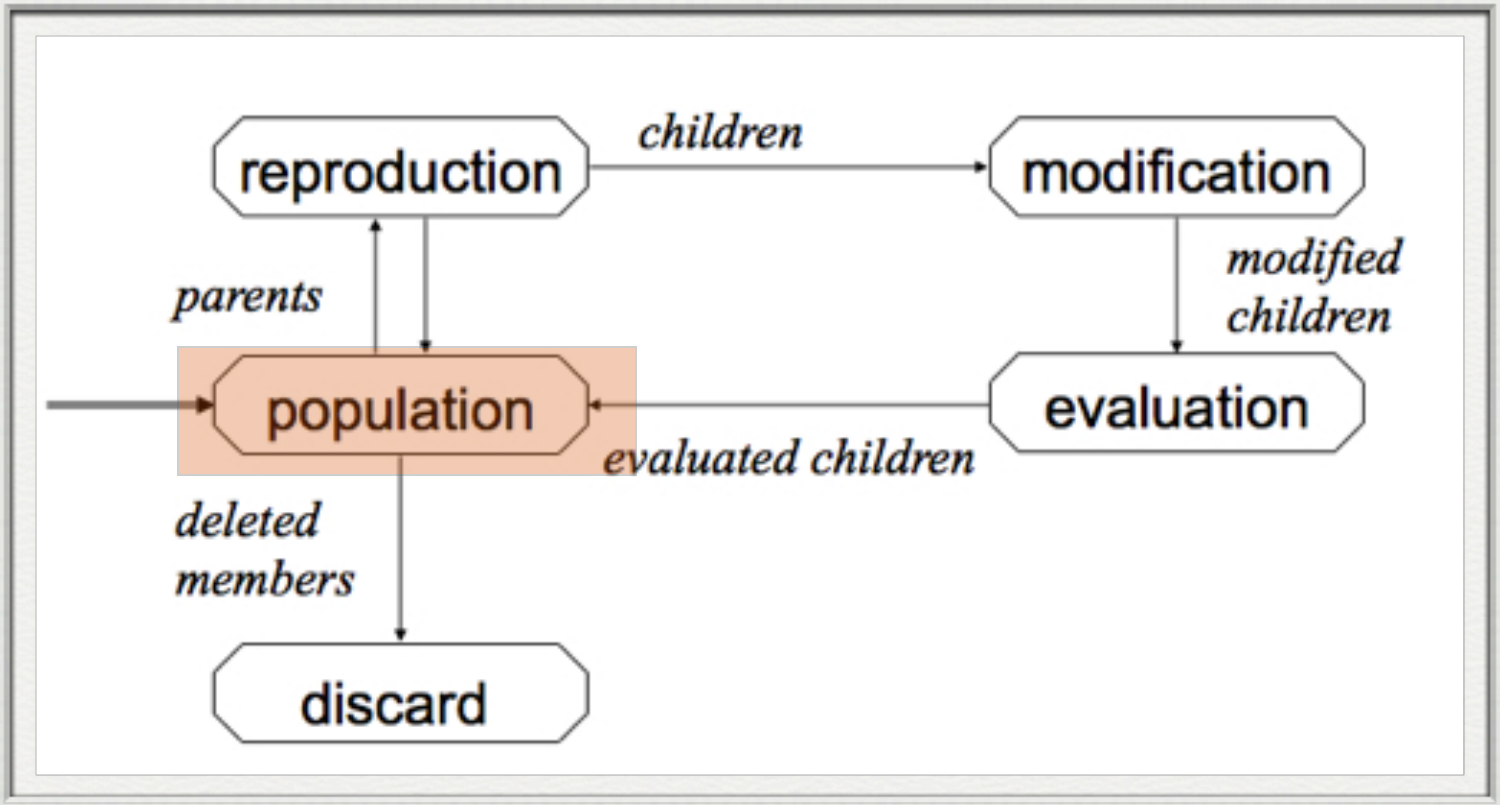
- A problem to solve, and ...

1. Encoding technique (gene, chromosome)
2. Initialization procedure (creation)
3. Evaluation function (environment)
4. Selection of parents (reproduction)
5. **Genetic operators (mutation, recombination)**
6. Parameter settings (practice and art)

SIMPLE ALGORITHM

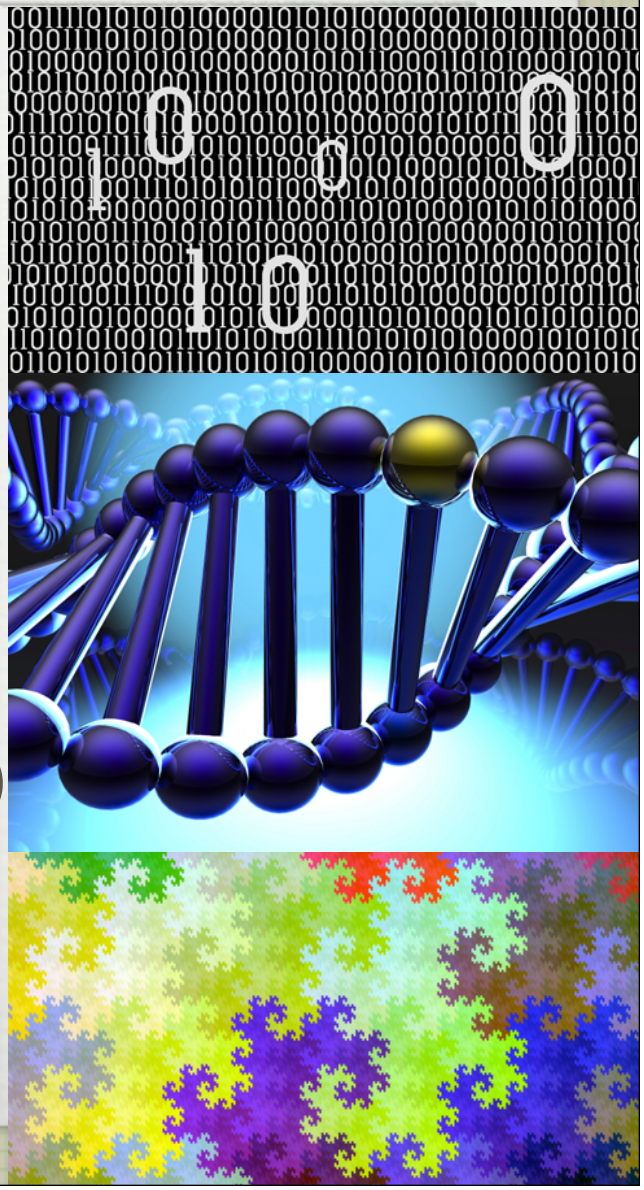


SIMPLE ALGORITHM

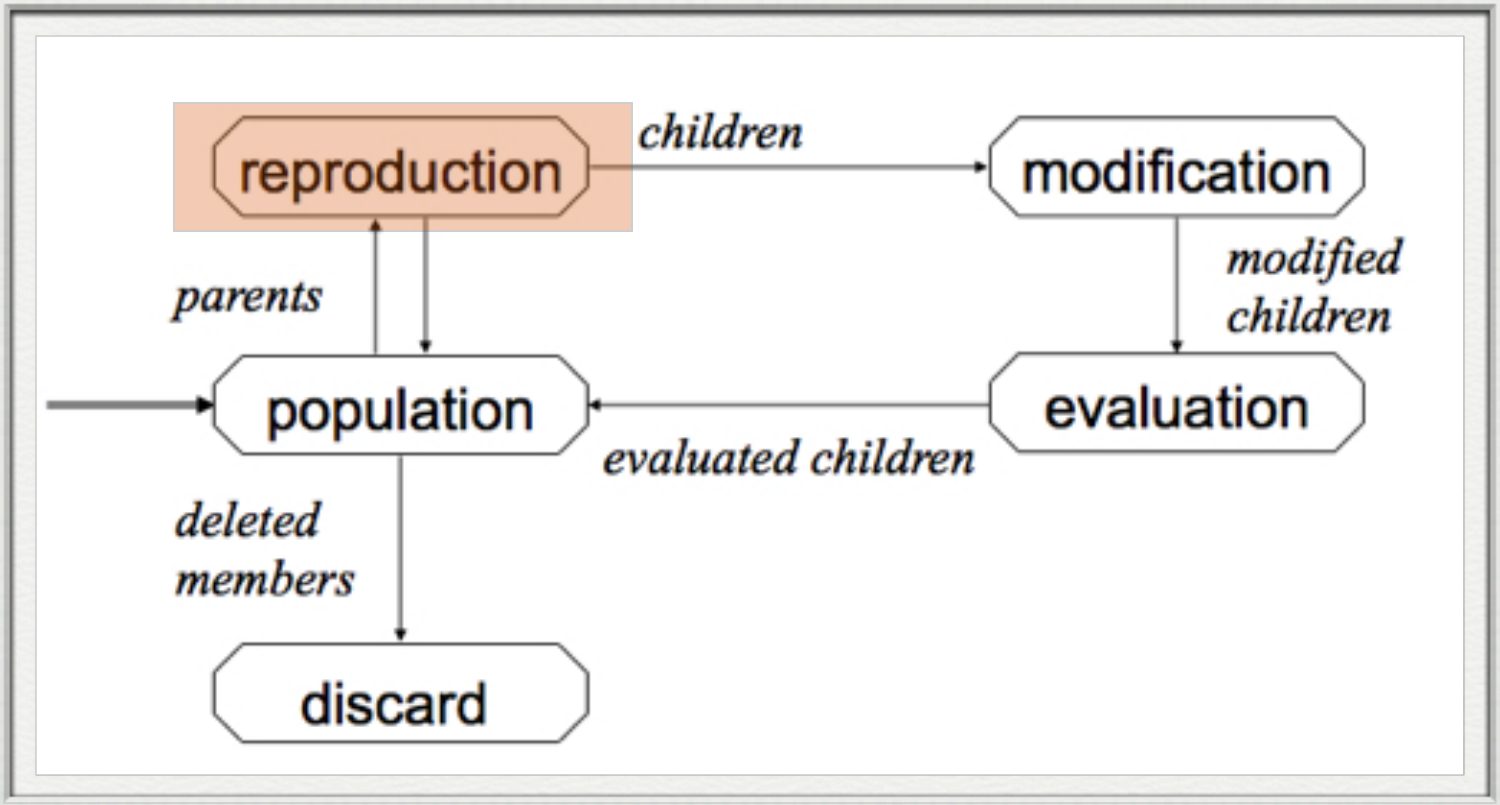


POPULATION (HOW WE ENCODE INFORMATION)

- **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 ...**

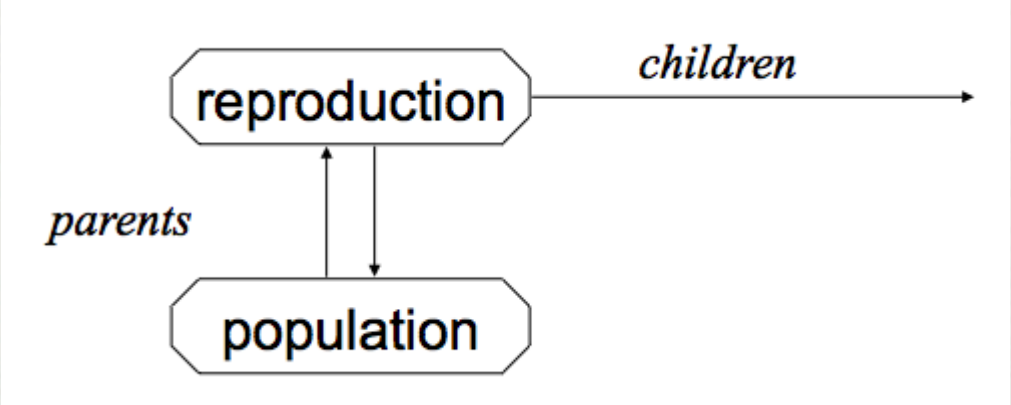


SIMPLE ALGORITHM

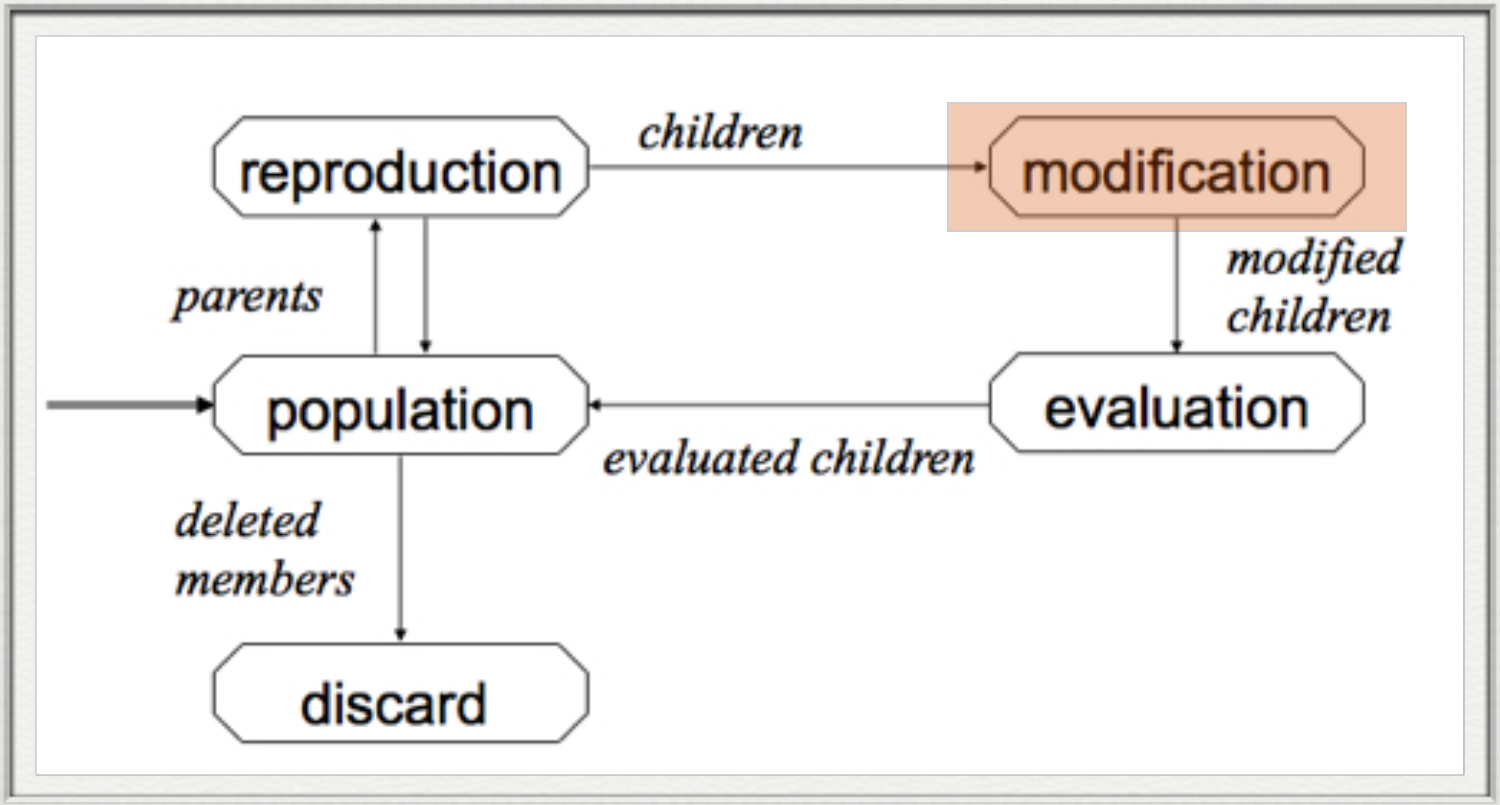


REPRODUCTION: CREATING NEXT GENERATION FROM A POPULATION?

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

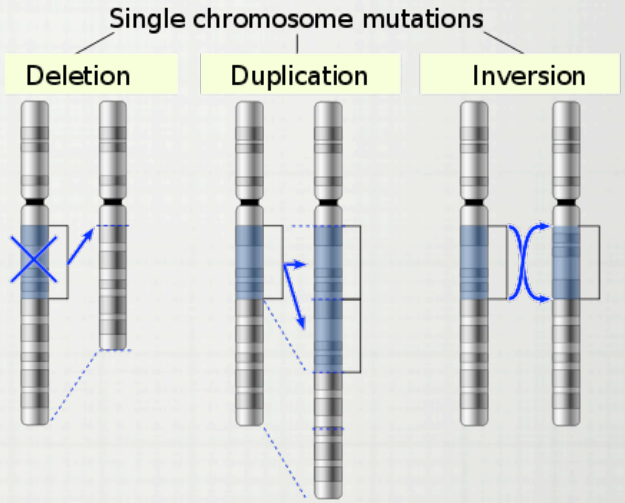


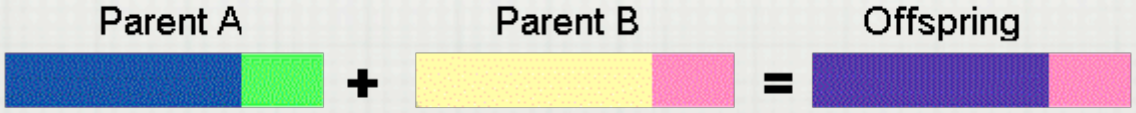
SIMPLE ALGORITHM

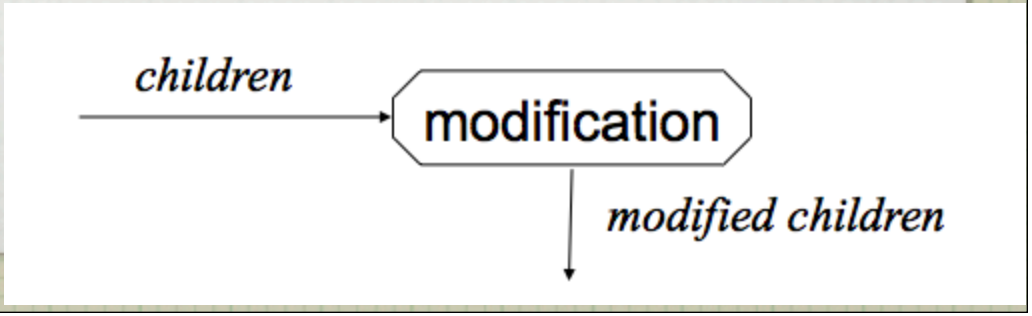


CHROMOSOME MODIFICATION

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



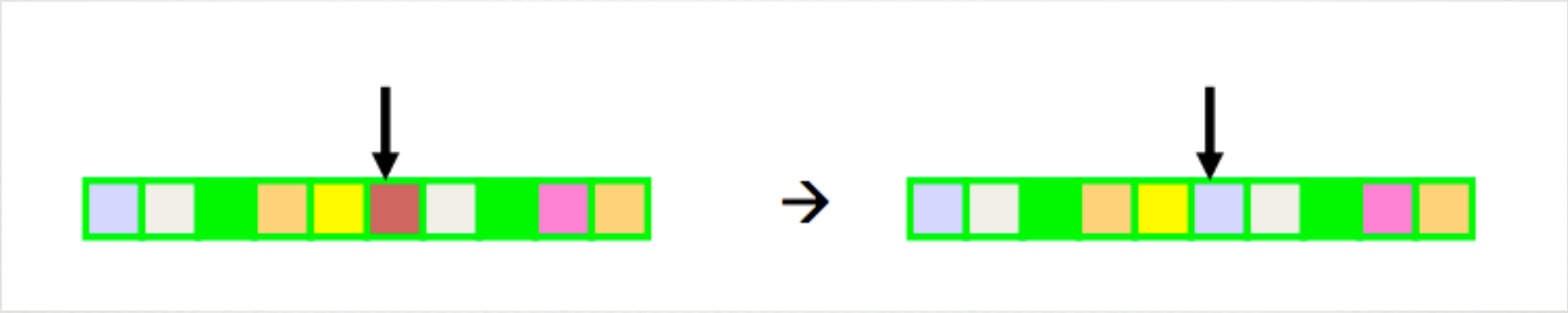




MUTATIONS
LOCAL MODIFICATIONS

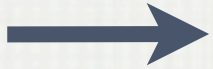
- **Example #1**
  - **Before**: (1 0 1 **1** 0 1 1 0)
  - **After**: (1 0 1 **0** 0 1 1 0)
- **Example #2**
  - **Before**: (1.38 **-69.4** 326.44 0.1)
  - **After**: (1.38 **-67.5** 326.44 0.1)
- **Mutations:**
  - Causes movement in the search space (local or global)
  - Restores lost information to the population

MUTATION: NEW OFFSPRING FROM SINGLE PARENT



CROSSOVER: RECOMBINATION

- **Simple crossover:**
  - **P1** (0 1 1 0 1 0 0 0)
  - **P2** (1 1 0 1 1 0 1 0)



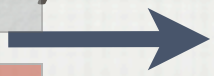
(0 1 0 1 1 0 1 0) **C1**

(1 1 1 0 1 0 0 0) **C2**

- **Crossover is a critical feature of genetic algorithms:**
  - **It greatly accelerates search early in evolution of a population**
  - **It leads to effective combination of schemata (subsolutions on different chromosomes)**

DOUBLE CROSSOVER

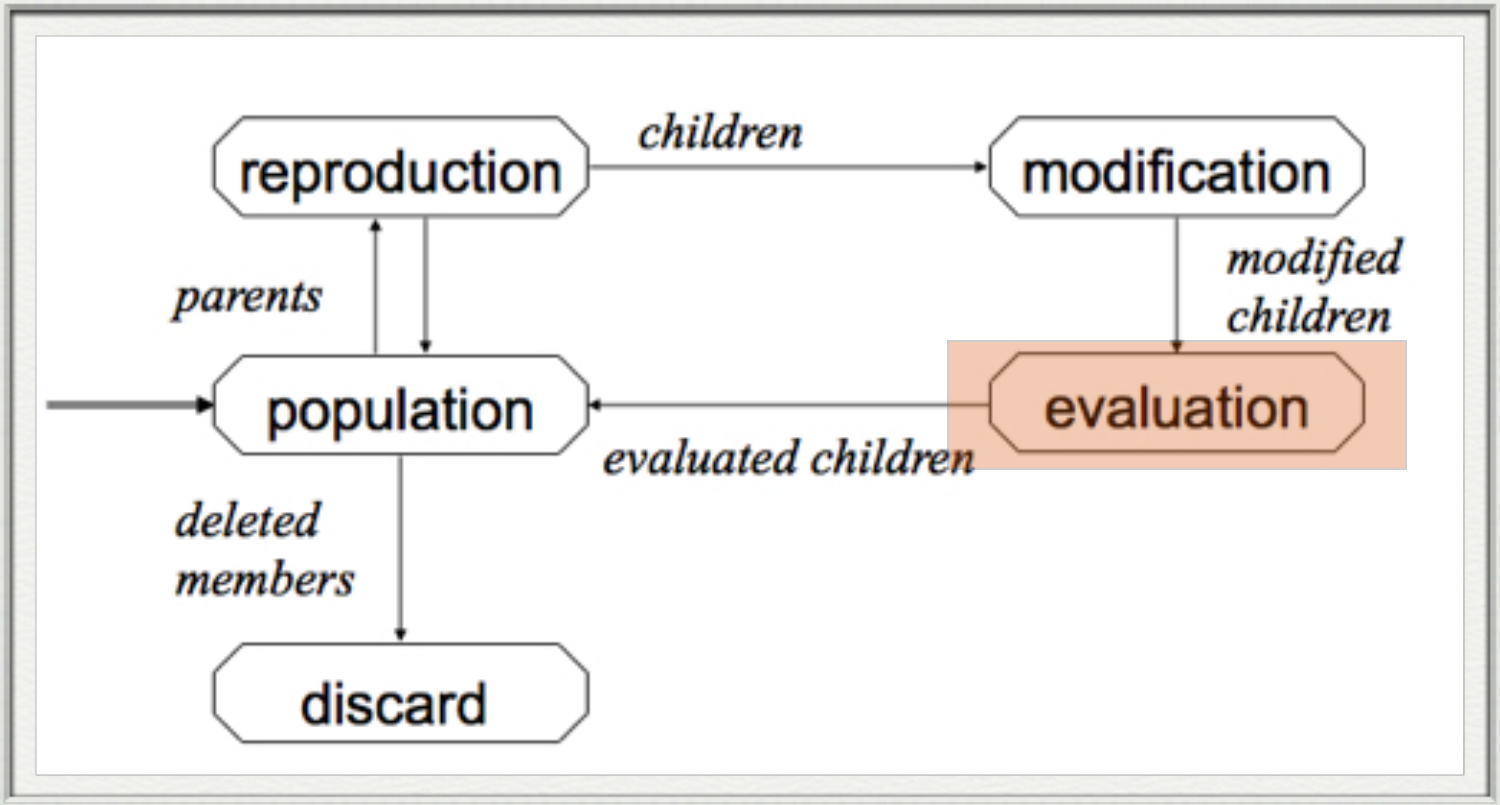
- P1 (0 1 1 0 1 0 0 0)
- P2 (1 1 0 1 1 0 1 0)



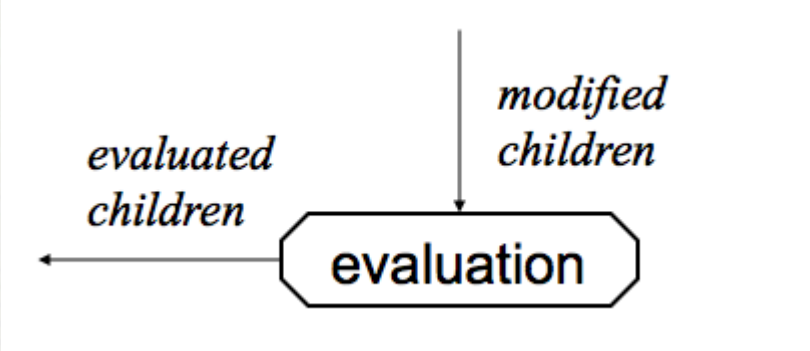
(0 1 0 1 1 0 0 0) C1

(1 1 1 0 1 0 1 0) C2

SIMPLE ALGORITHM

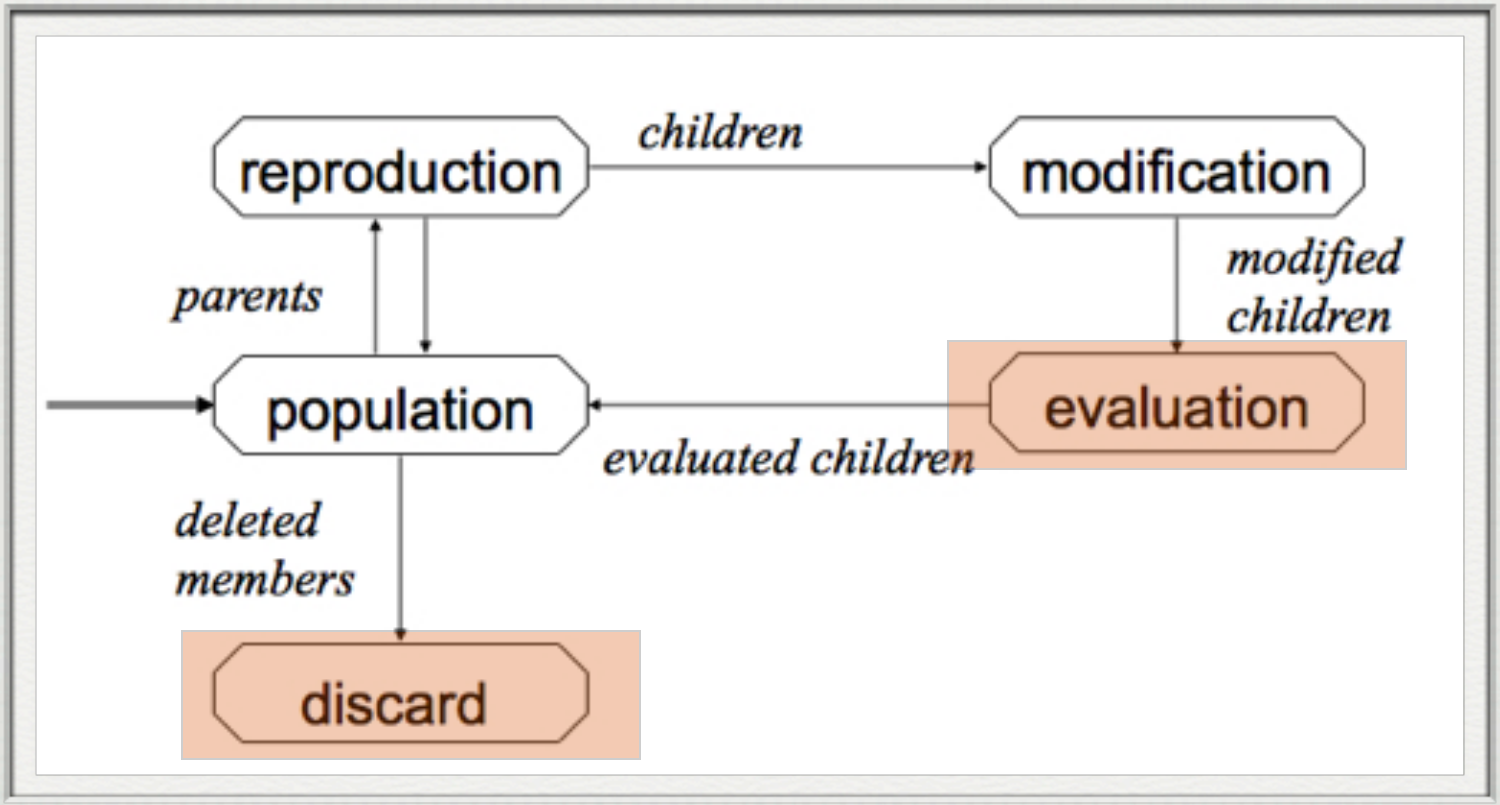


EVALUATION: "FITNESS"

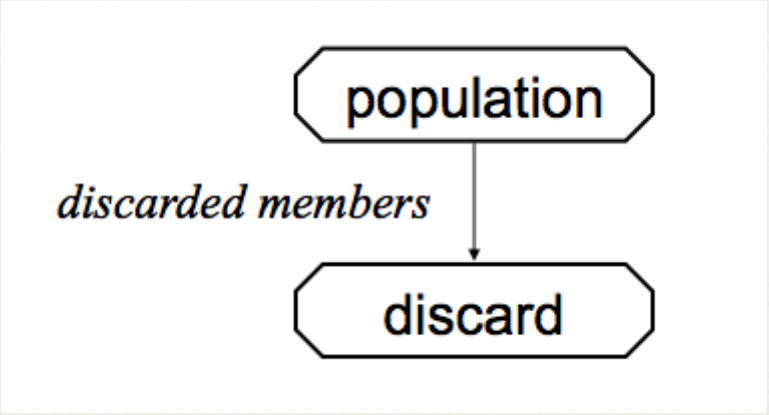


- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving
- **Example:** energy function, etc.

SIMPLE ALGORITHM



DELETION

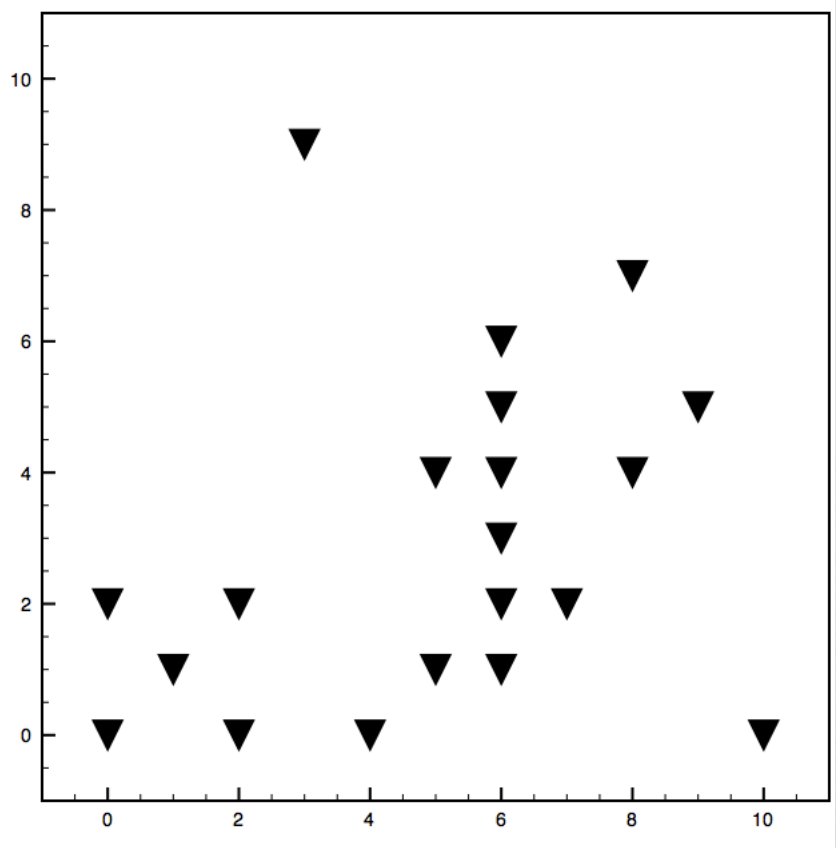


- ***Generational* GA:**
  entire populations replaced with each iteration
- ***Steady-state* GA:**
  a few members replaced each generation

FIRST EXAMPLE: SALESMAN

BACK TO THE TRAVELING SALESMAN

- We have already solve this problem using simulated annealing
- Shortest path across the cities?

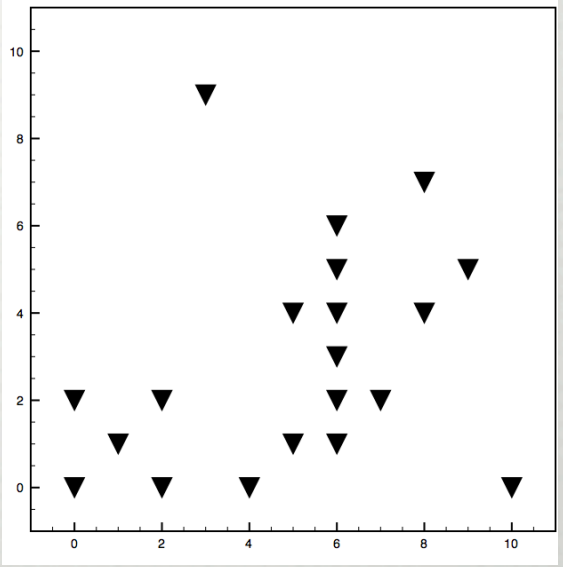


ELEMENTS OF THE ALGORITHM

1. **Encoding technique (gene, chromosome)**
2. **Initialization procedure (creation)**
3. **Evaluation function (environment)**
4. **Selection of parents (reproduction)**
5. **Genetic operators (mutation, recombination)**

ELEMENTS OF THE ALGORITHM: **ENCODING**

- **Encoding technique (gene, chromosome)**
  - each chromosome of the population is made up of an ordered list of cities (integer)
  - e.g. (4,1,5,0,2,3) or (1,2,5,0,4,3) are two possible member of the population
- **Here: ordered list of all cities is one possible element**

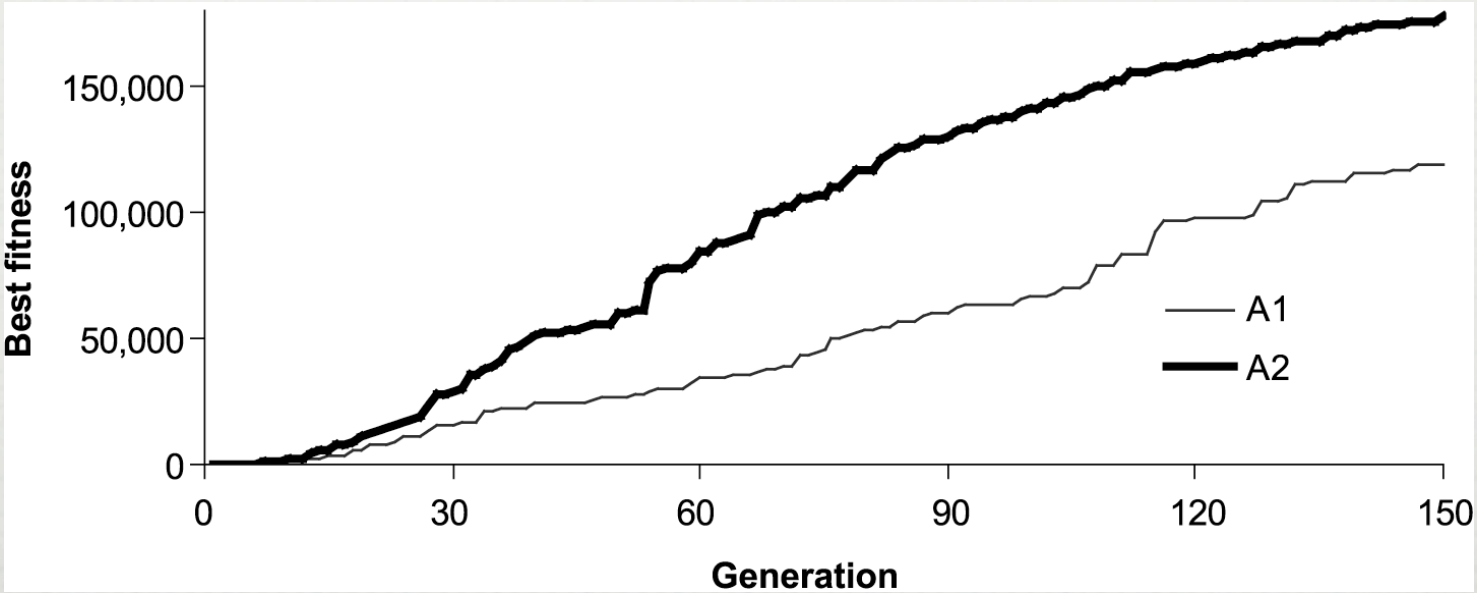


ELEMENTS OF THE ALGORITHM: **INITIALIZATION**

- **Initialization procedure (creation)**
  - **our old friend the random number generator**
  - **for example using a shuffle operation form (0,1,2,3,4,5)**

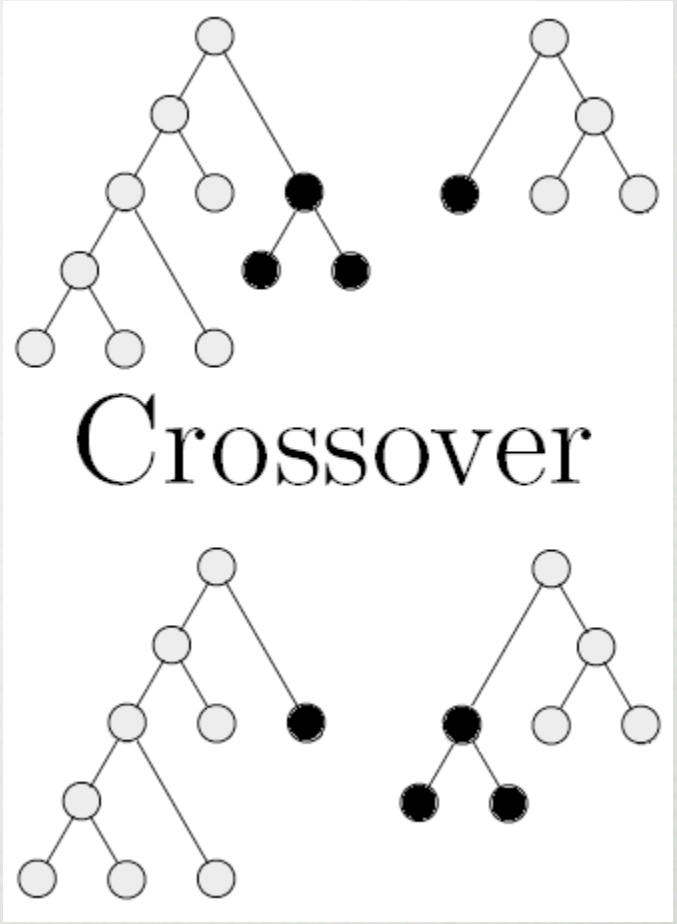
ELEMENTS OF THE ALGORITHM:
EVALUATION FUNCTION

- Evaluation function (environment)
  - f(x)-> total distance to travel across the given sequence



ELEMENTS OF THE ALGORITHM: **REPRODUCTION**

- **Selection of parents (reproduction)**
  - **sort the parents by fitness, discard worst half of the population**
  - **pick two parents randomly; then crossover**

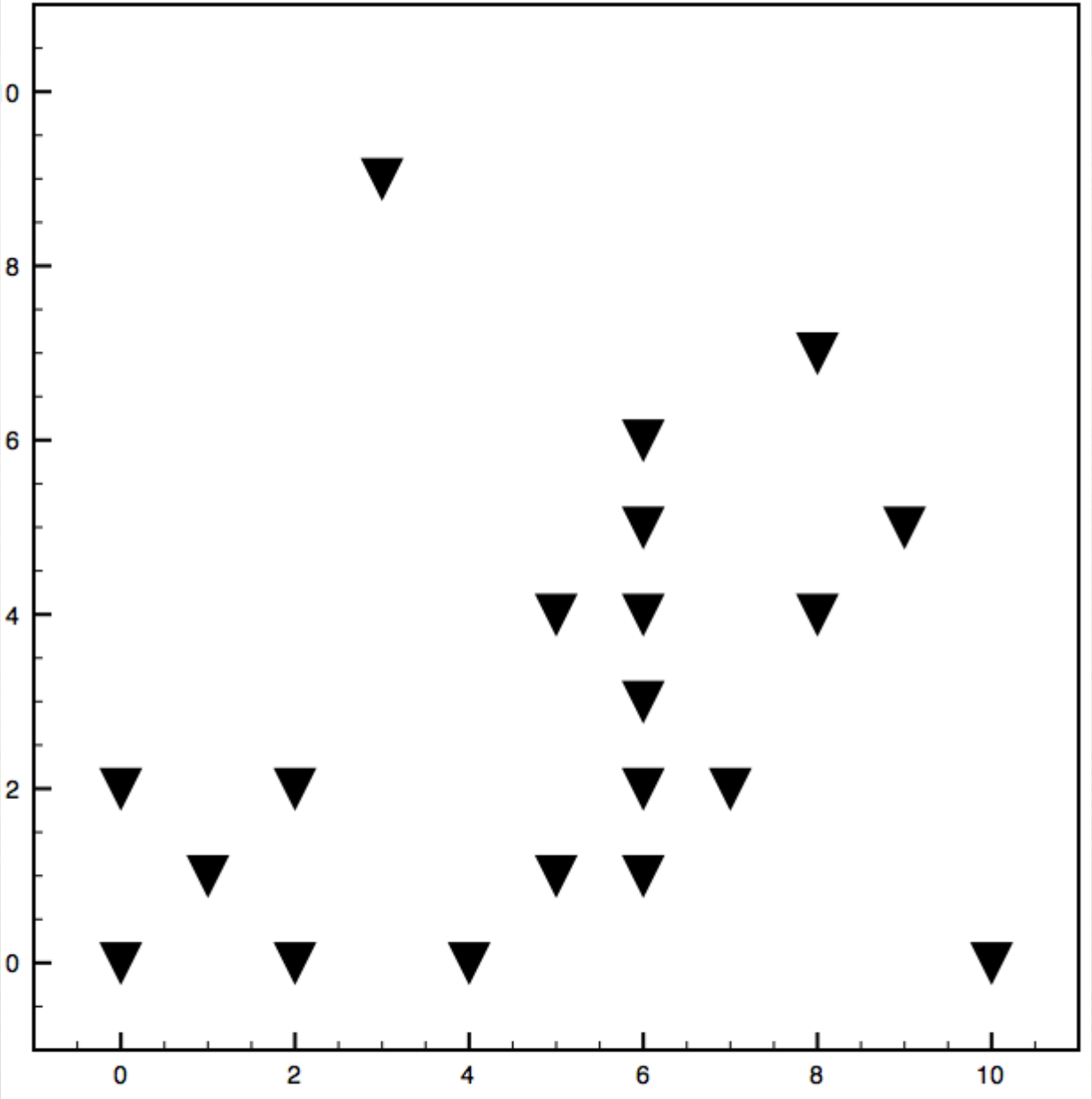


CROSSOVER (ORDER I CROSSOVER)

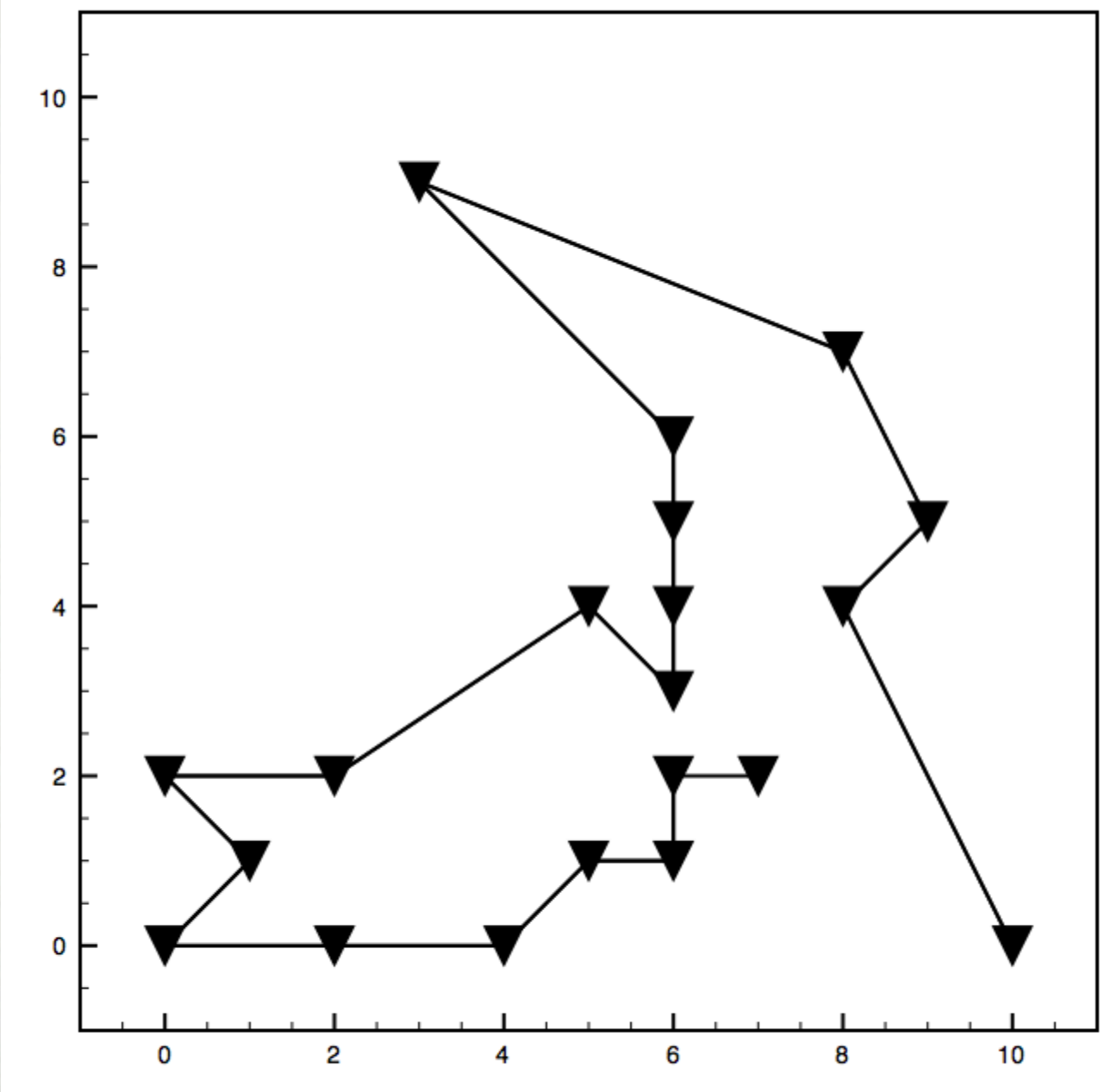
- **Crossover combines inversion and recombination:**
  - **Parent 1** (3 5 7 2 1 6 4 8)
  - **Parent 2** (2 5 7 6 8 1 3 4)
  - **Child 1** (5 8 7 2 1 6 3 4)
  - **Child 2?** (5 2 7 6 8 1 4 3)

ELEMENTS OF THE ALGORITHM:
GENETIC OPERATIONS

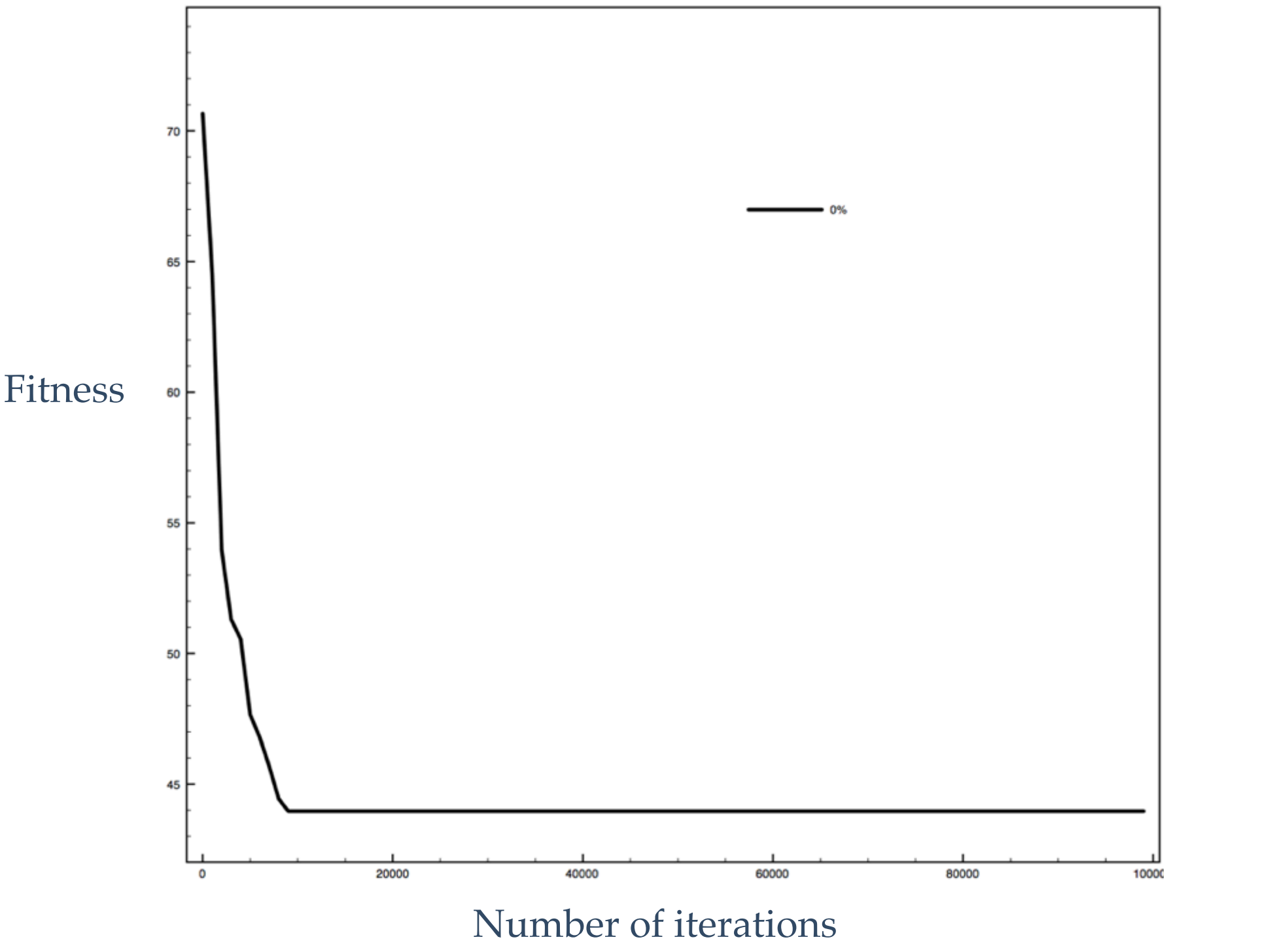
- Genetic operators (mutation, recombination)
- Mutation involves reordering of the list:
  - Before: (5 8 7 2 1 6 3 4)
  - After: (5 8 6 2 1 7 3 4)
- The positions are picked randomly


0
8
6
4
2
0
0
2
4
6
8
10

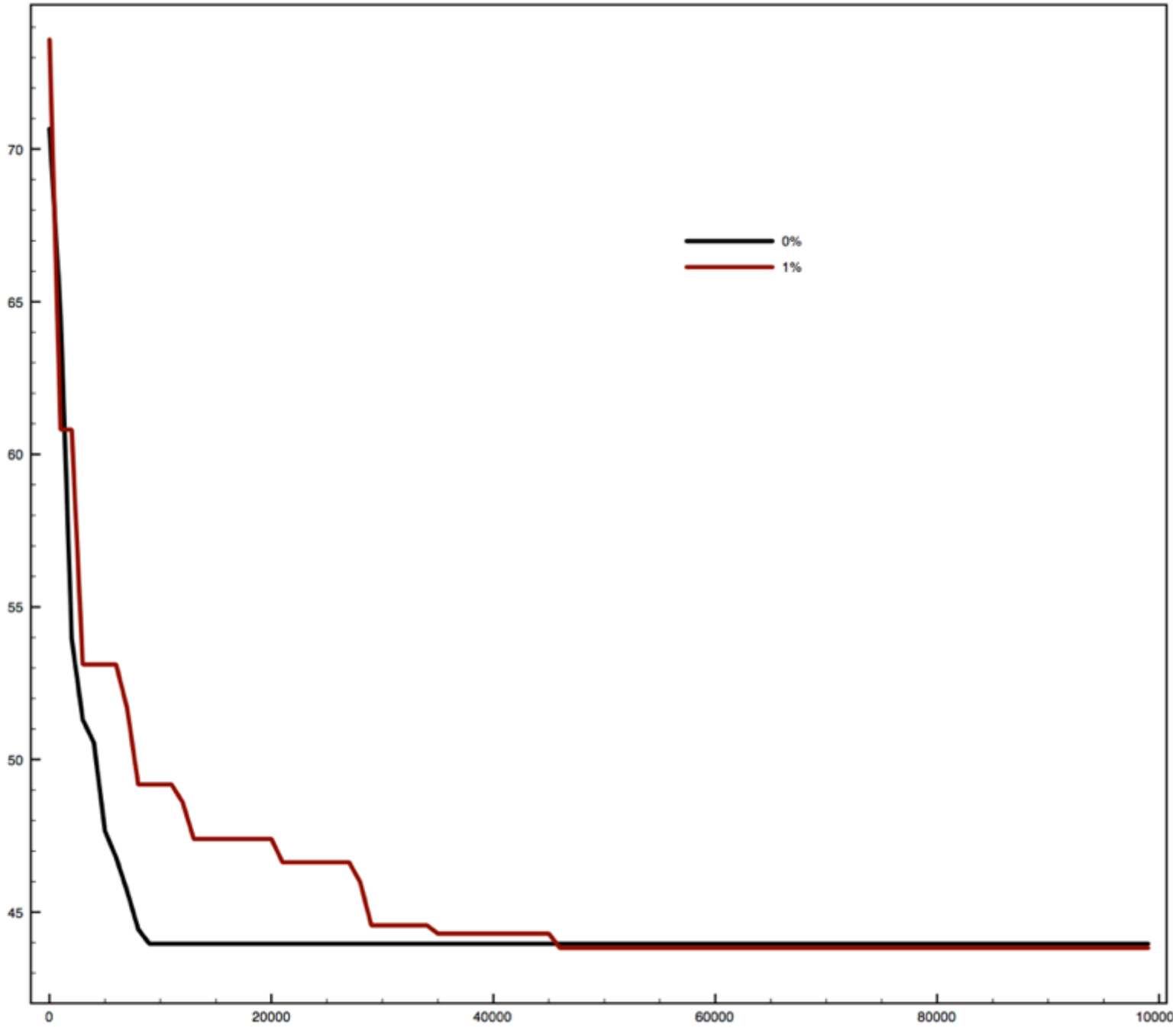
CONVERGED SOLUTION



IMPORTANCE OF PARAMETERS: 1. MUTATIONS

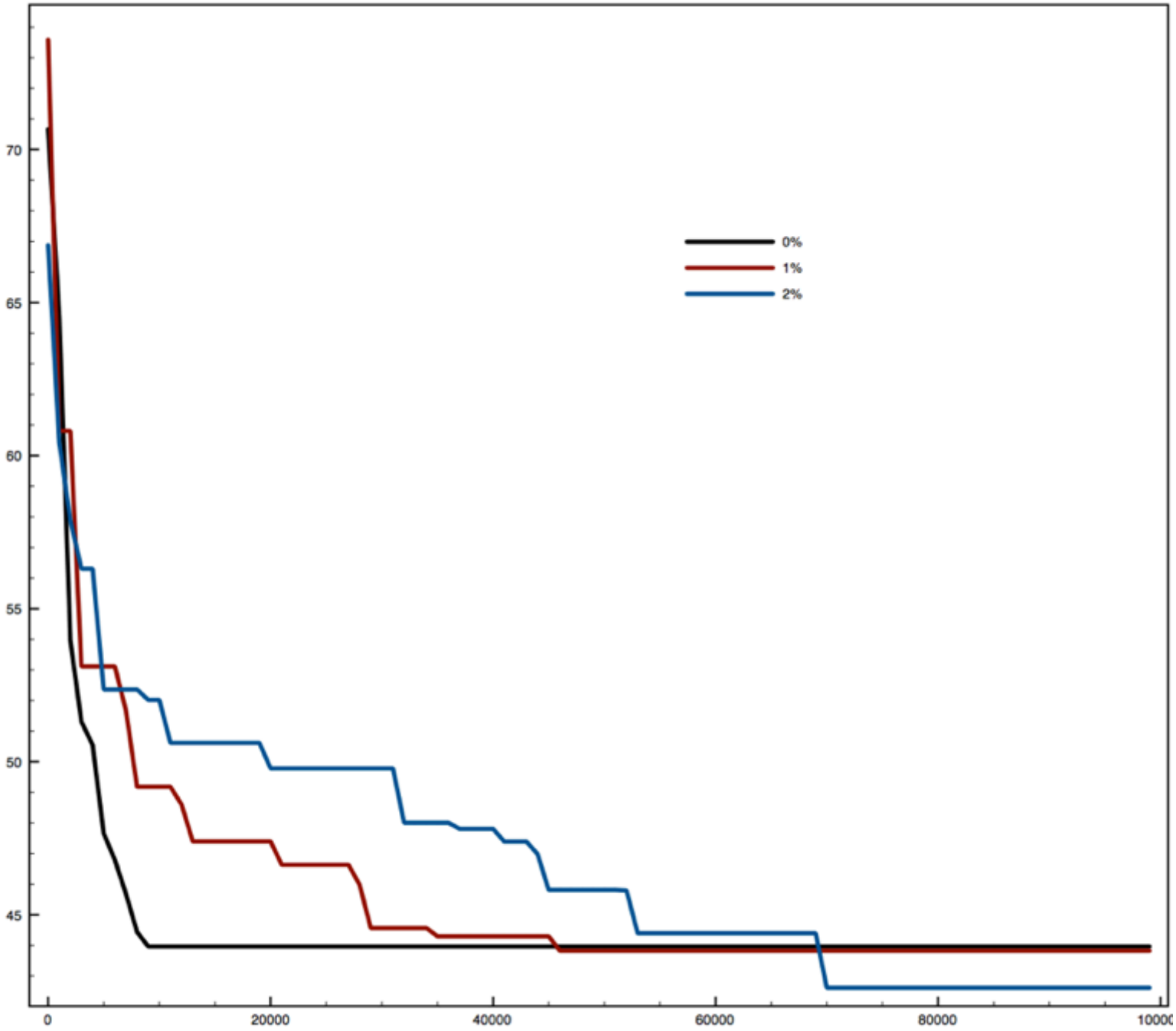

Fitness
70
65
60
55
50
45
0%
0
20000
40000
60000
80000
10000
Number of iterations

Fitness



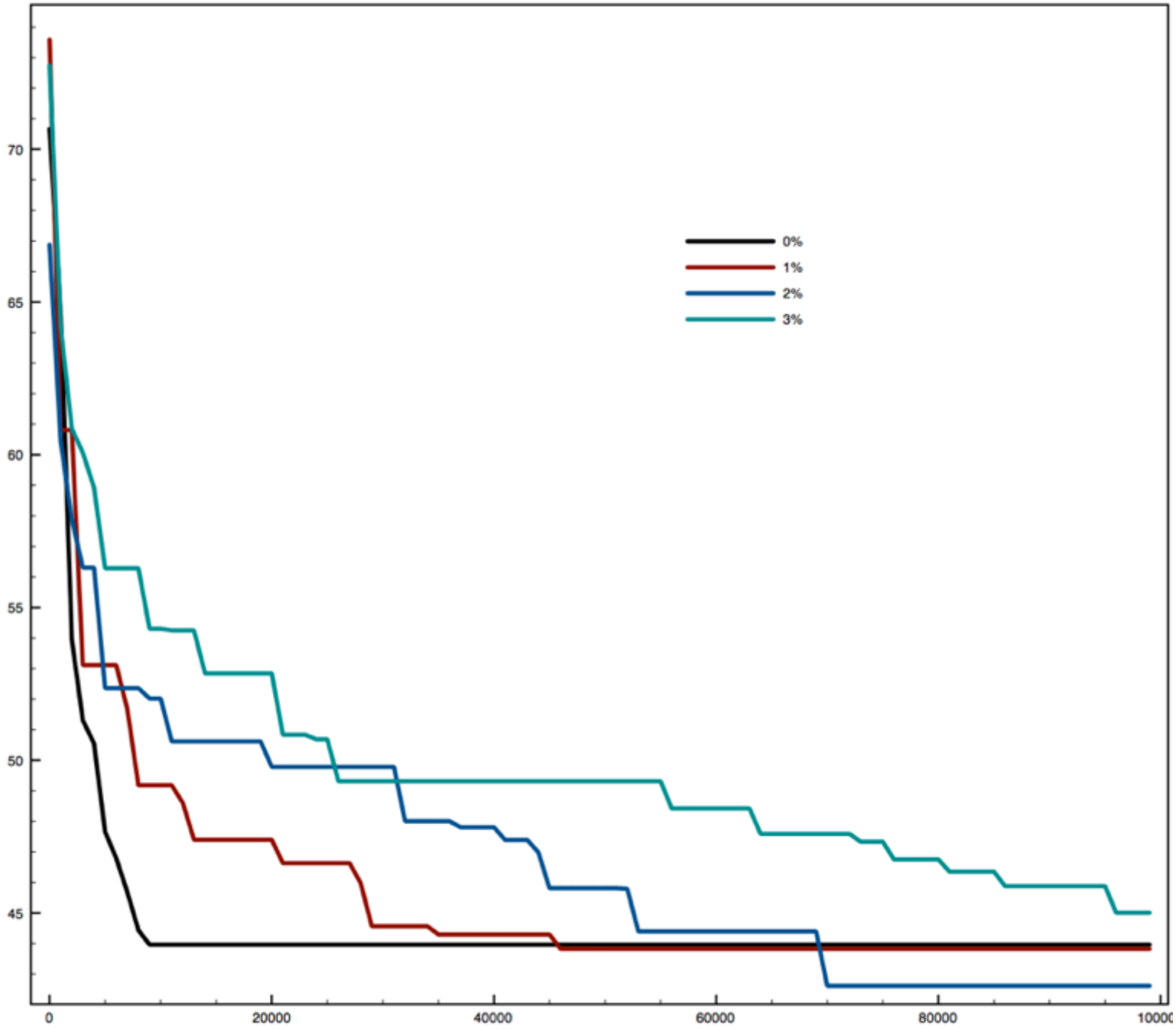
Number of iterations

Fitness



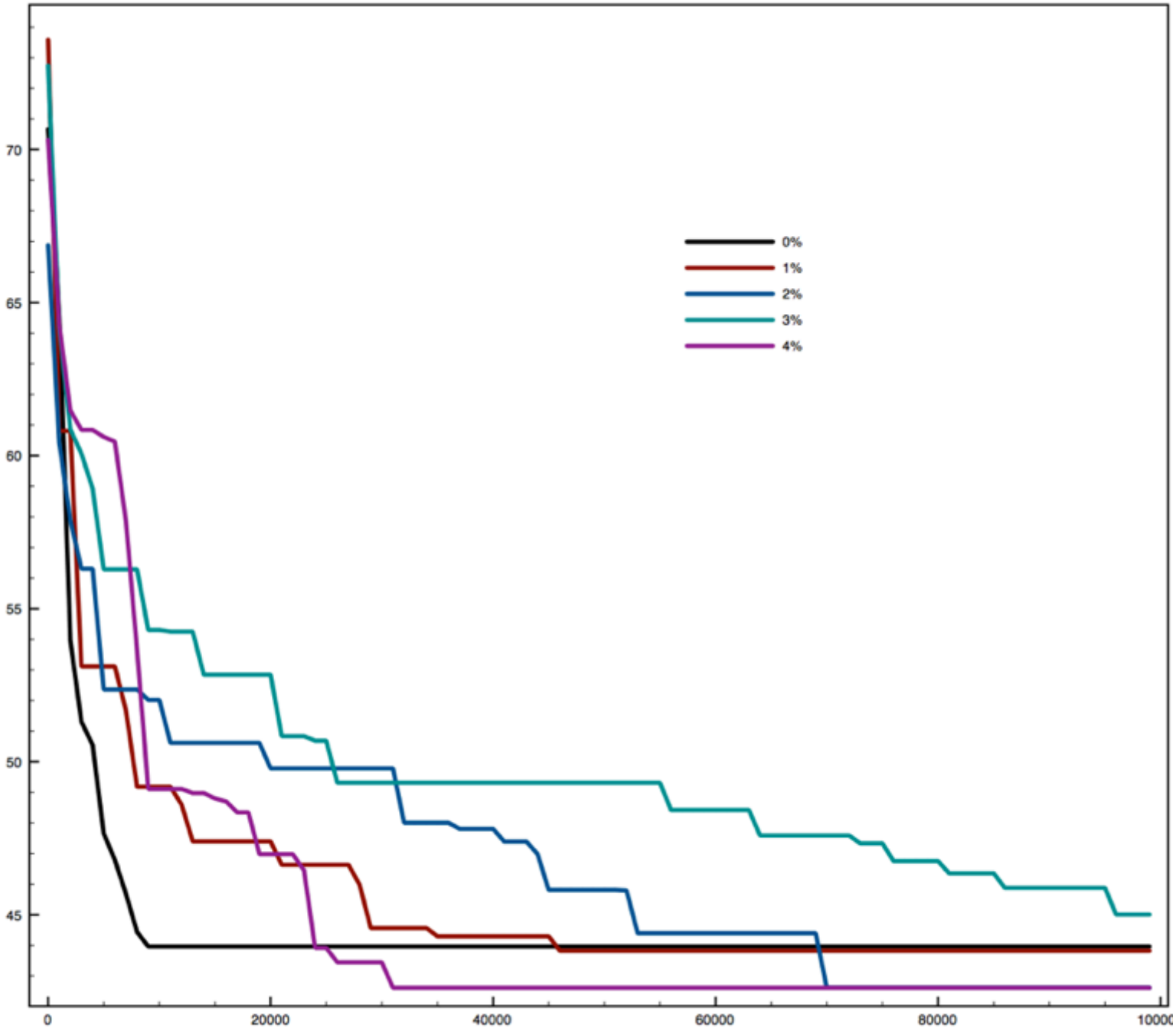
Number of iterations

Fitness



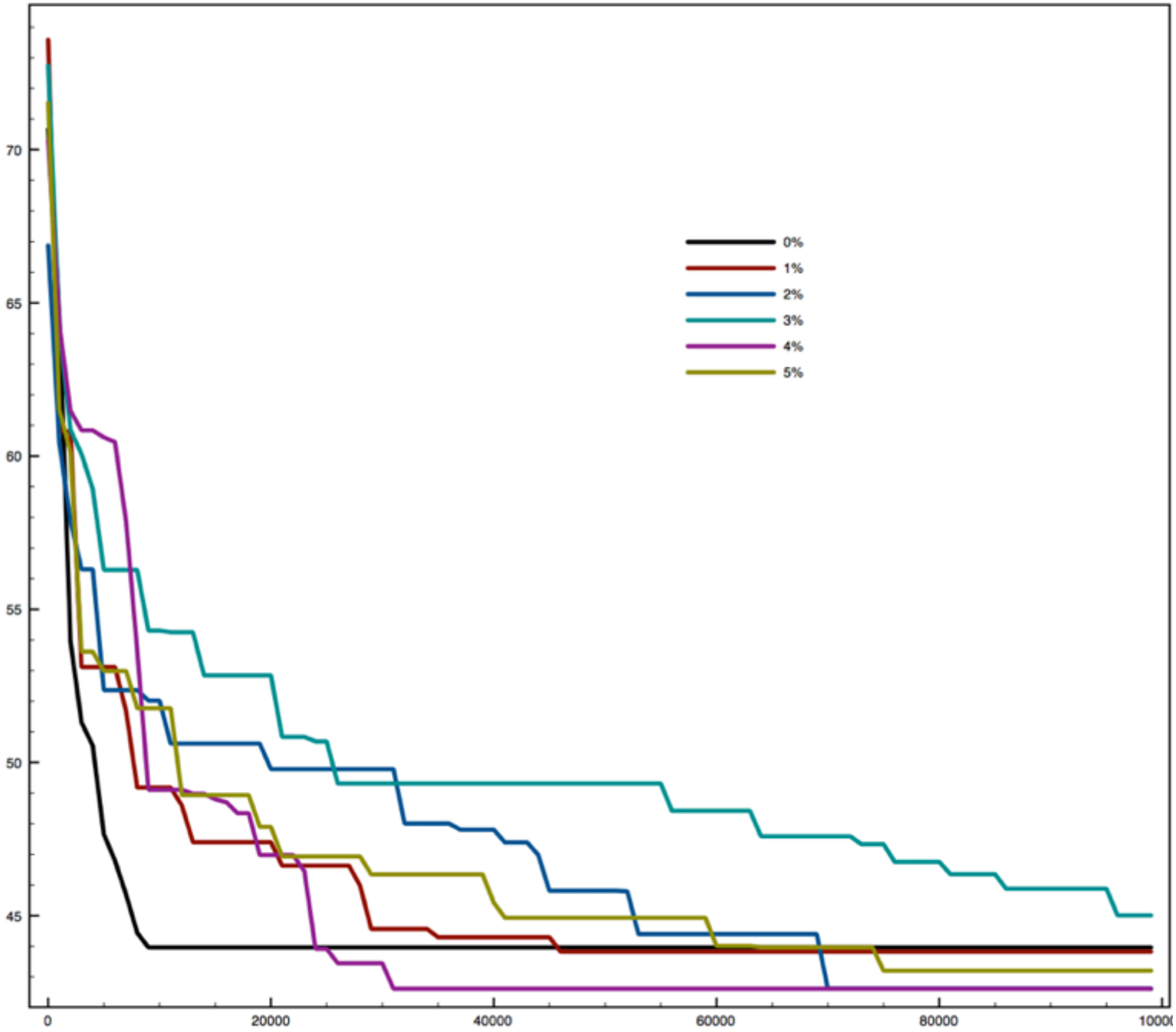
Number of iterations

Fitness



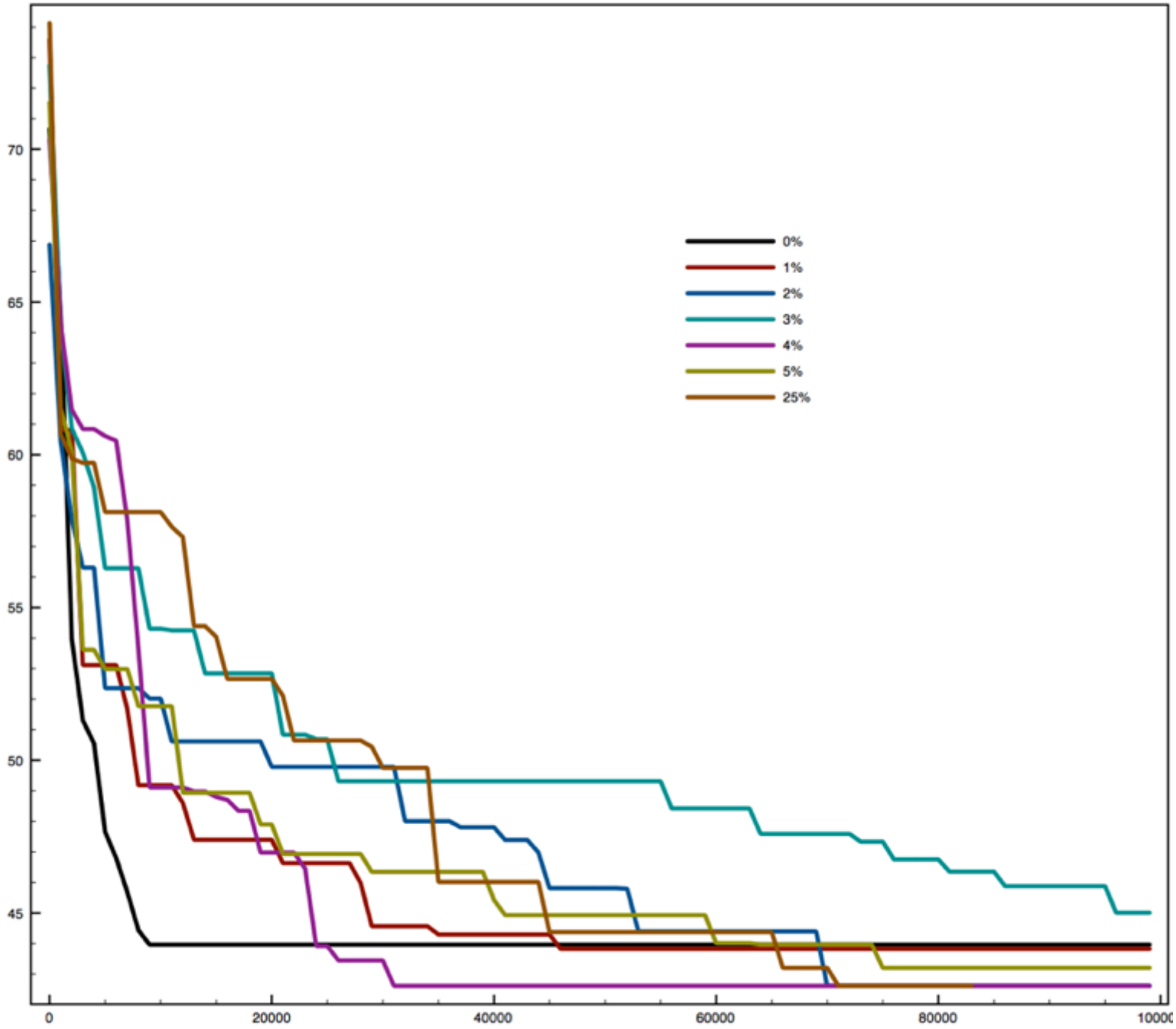
Number of iterations

Fitness



Number of iterations

Fitness

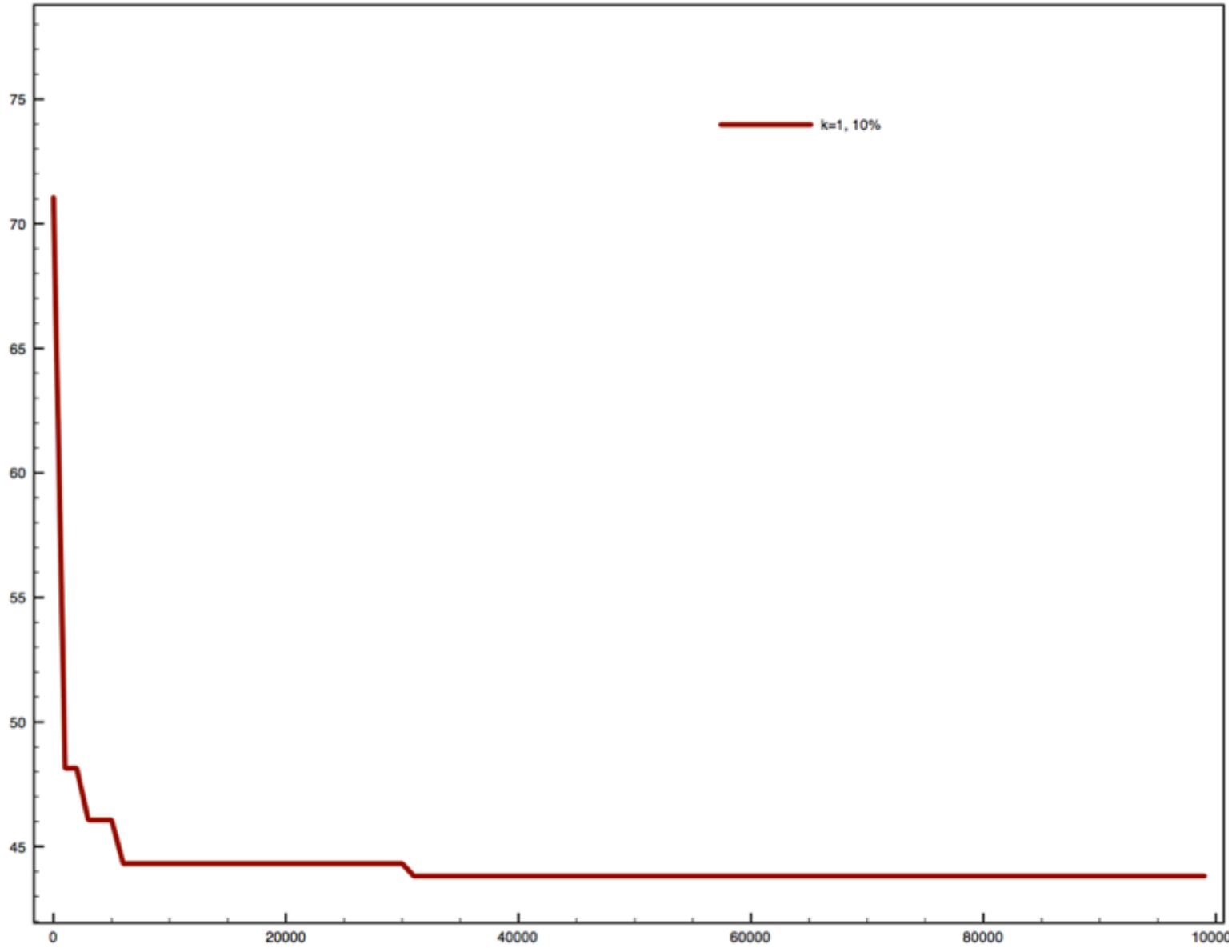


Number of iterations

CONCLUSION: ENOUGH MUTATIONS BUT NOT TOO MANY!

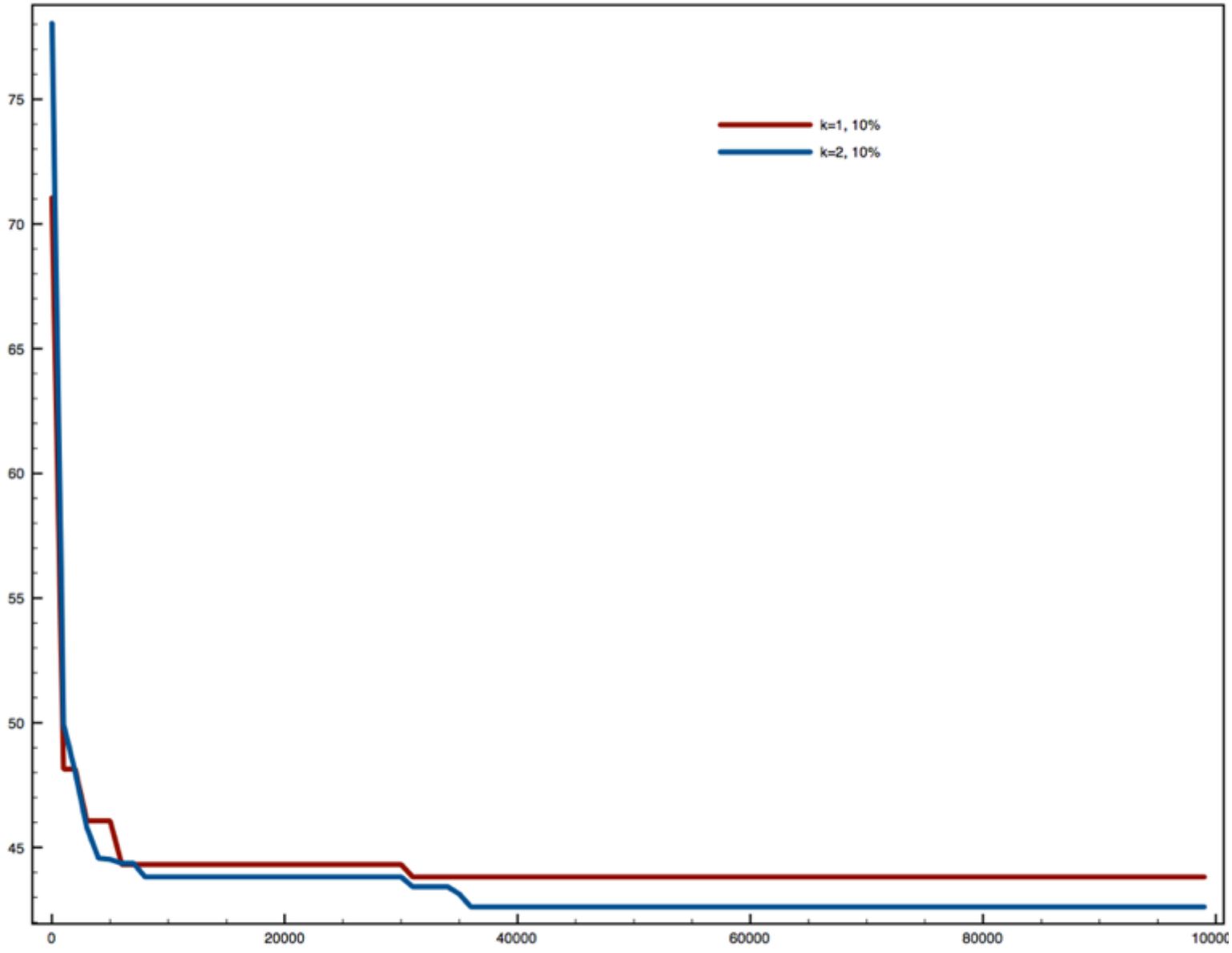
IMPORTANCE OF PARAMETERS: 2. KEEPING OLD GENERATIONS

Fitness



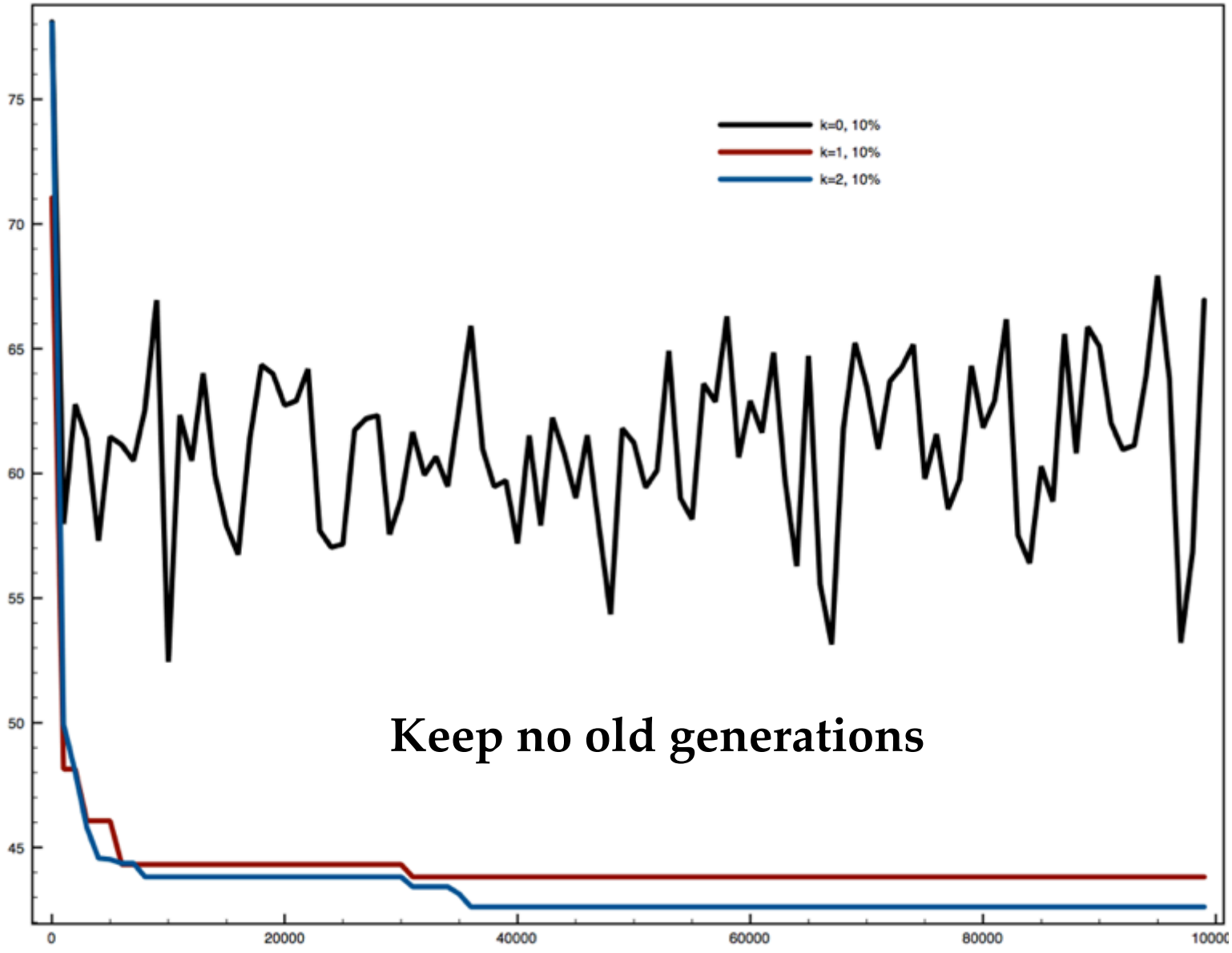
Number of iterations

Fitness



Number of iterations

Fitness



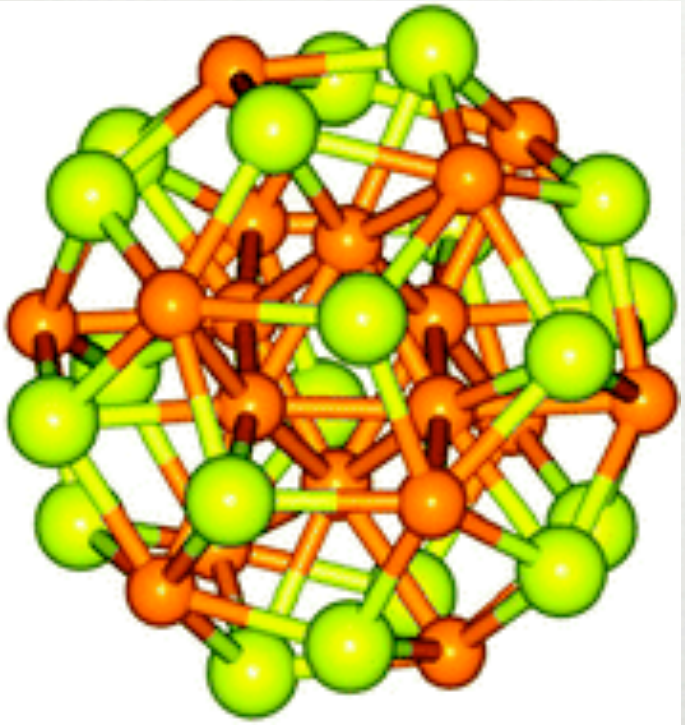
Keep no old generations

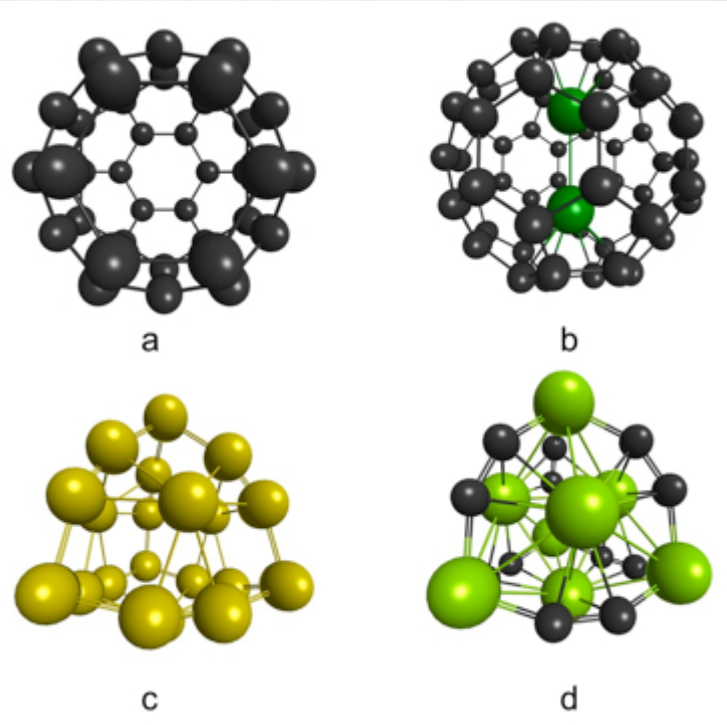
Number of iterations

SECOND EXAMPLE: NANOCLUSTERS

PROBLEM

- **How can you arrange N identical charges on a sphere in such a way as to minimize the electrostatic energy?**





ELEMENTS OF THE ALGORITHM

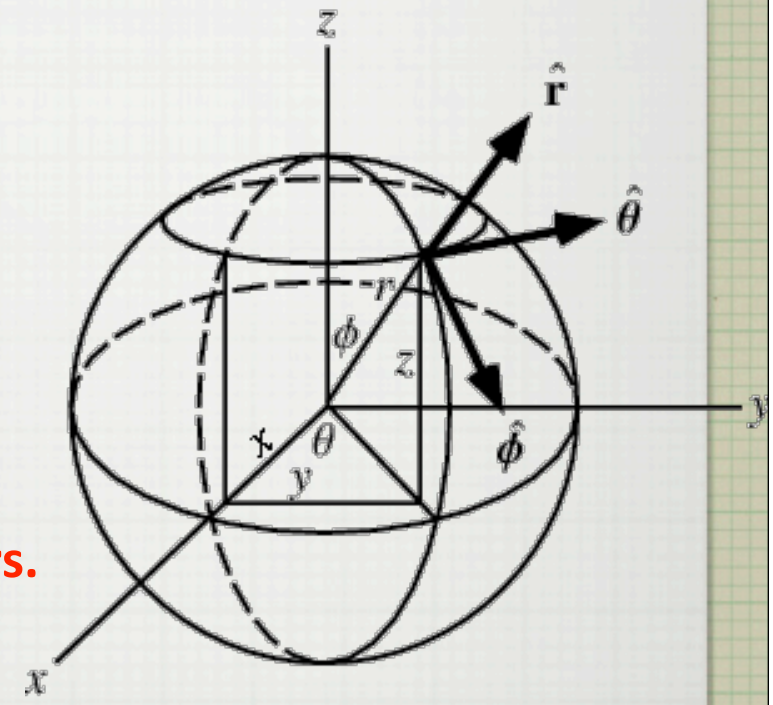
- **Encoding technique (gene, chromosome)**
- **Initialization procedure (creation)**
- **Evaluation function (environment)**
- **Selection of parents (reproduction)**
- **Genetic operators (mutation, recombination)**

ENCODING

WHAT DO WE NEED TO ENCODE? (HOW TO CREATE A CHROMOSOME?)

- **Spherical Coordinates:**

$$
\begin{aligned}
x &= r\sin\theta\cos\phi \\
y &= r\sin\theta\sin\phi \\
z &= r\cos\theta
\end{aligned}
$$



- **2 angles per charge (r is fixed) → 2 real numbers.**
- **Encoding:** we will code in binary form
  - remember lecture 1 on how to code fractional numbers
  - the number of bits we use per number is a parameter

EXAMPLE OF 2 NUMBERS, USING 8 BITS

- **Two angles (reduced in the [0,1] range) picked randomly:**
  - **0.67637657932937145233**
  - **0.86117408191785216331**
- **Corresponding Gene (8 bits):**
  - **10101101110111 00**
  - **10101101110111 00**
- **10101101: 0.5 + 0.125 + 0.03125 + 0.015625 +0.00390625**
- **11011100: ?**

FROM LECTURE 1: HOW TO CONVERT THE FRACTIONAL PART OF A NUMBER FROM BASE 10 TO BASE 2 ?

- **Repeatedly double the number to be converted, record if the result is at least 1, and then throw away the integer part.**
- **Example:** $(1/3)_{10}$
  - **Step 1: 1/3*2 < 1 -> 0**
  - **Step 2: 2/3*2 >1 -> 1 (remove 1, work on 1/3)**
  - **Step 3 : 1/3*2 < 1 -> 0**
  - **...**
  - $(1/3)_{10}=(0.0101010101010101...)_2$
- **Example:** $(0.1)_{10}$
  - $(0.0001100110011001100...)_2$
  - **repeating binary fraction!**

EXAMPLE OF ENCODING

```
void encode(VecDoub_I &x, VecInt_O &dna){
        int nb=dna.size()/x.size(); //number of byte per x value
        for(int i=0;i<x.size(); i++){
                double d=2;
                double sum=x[i];
                dna[nb*i]=(int)(0.5+sum);
                for(int j=1;j<nb;j++){
                        sum-=(double) (dna[nb*i+j-1])/d;
                        dna[nb*i+j]=(int) (0.5+d*sum);
                        d*=2;
                }
        }

}
```

INITIALIZATION PROCEDURE

STARTING CHROMOSOMES AND GENES

1. **Use random number generator**
2. **Create more genes than needed**
3. **Sort them in ascending order of fitness**
4. **Discard the surplus from the bottom**

EVALUATION: FITNESS

FITNESS

Pairwise electrostatic energy (with N electrons)

$$E = \sum_{i=1}^{N} \sum_{j \neq i}^{N} \frac{1}{r_{ij}}$$

SELECTION OF PARENTS

SELECTION

- **Survival of the fittest**
- **Parents are chosen randomly “from the chosen one”**

GENETIC OPERATIONS

CHILDREN GENERATION: SINGLE CROSSOVER

- We use single or multiple cross-over
  - Parents are picked randomly
  - in example here: (10 bits per gene, 2 variables)
- Example:
  - Parent 1: 1010|0000011101111
  - Parent 2: 1110|1100111010101
  - Child 1: 1110|0000011101111
  - Child 2: 1010|1100111010101

CHILDREN GENERATION: DOUBLE CROSSOVER

- We use single or multiple cross-over
  - Parents are picked randomly
  - (10 bits per gene, 2 variables)
- Example:
  - Parent 1: 1011010011111010
  - Parent 2: 1111111000111010 1
  - Child 1: 1111110001110101
  - Child 2: 1011010011111010

MUTATION

- We picked randomly a gene and we flip its value from 0 to 1 or 1 to 0
- We impose some of the fittest to be immune to mutation
- Example #1:
  - 1100011111111001
  - 1100011111111000
- Example #2:
  - 0010101000011101
  - 0000101000011101

FREE PARAMETERS

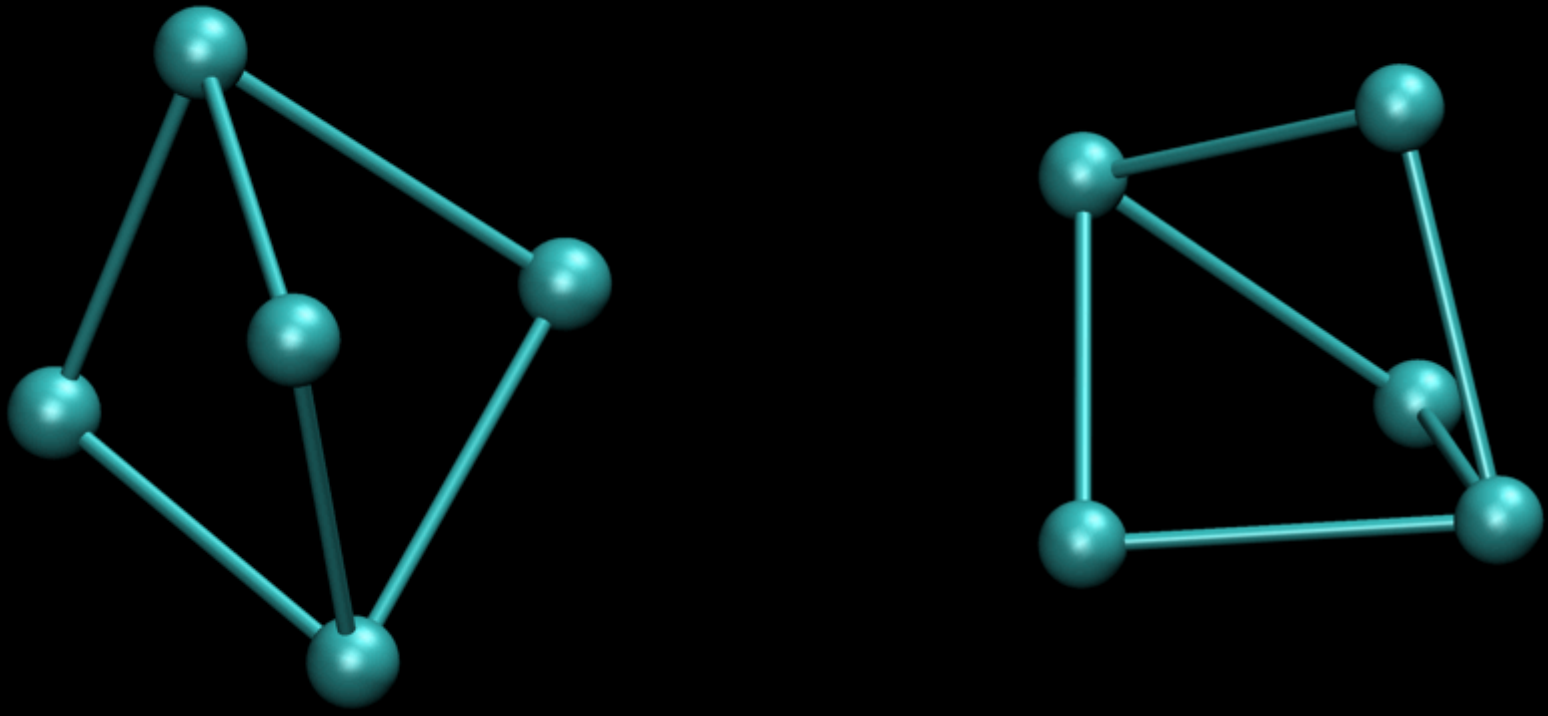
- number of bits per number
- number of mutations per generation
- number of chromosome immune to mutation
- number of crossover during recombination

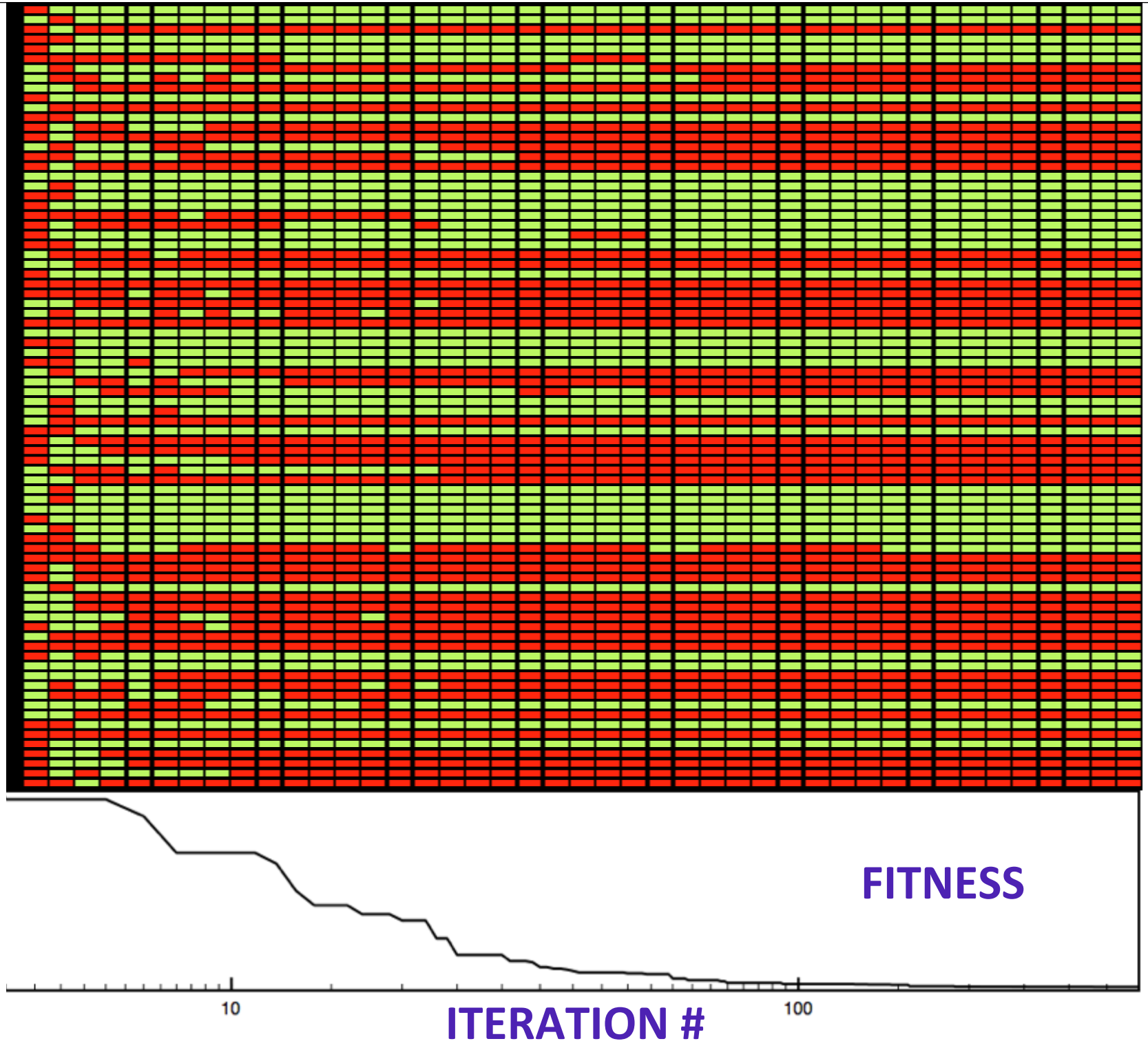
- some will help scan faster but sometimes at the expense of missing the true minimum
- there is no single good recipe

EXAMPLE: N=5

- N_B=8
- NCROSS=10
- $nmut_0$=10
- NMUT=5%
- sample size: 32
- 10,000 generations

CONVERGED SOLUTION

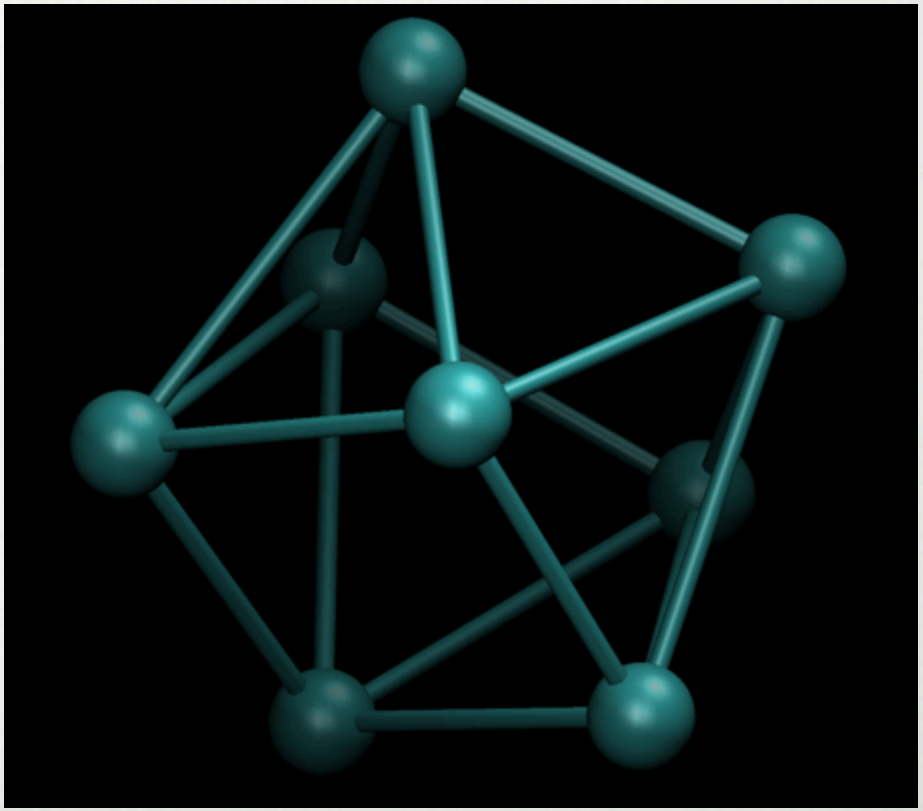



FITNESS
10
100
ITERATION #

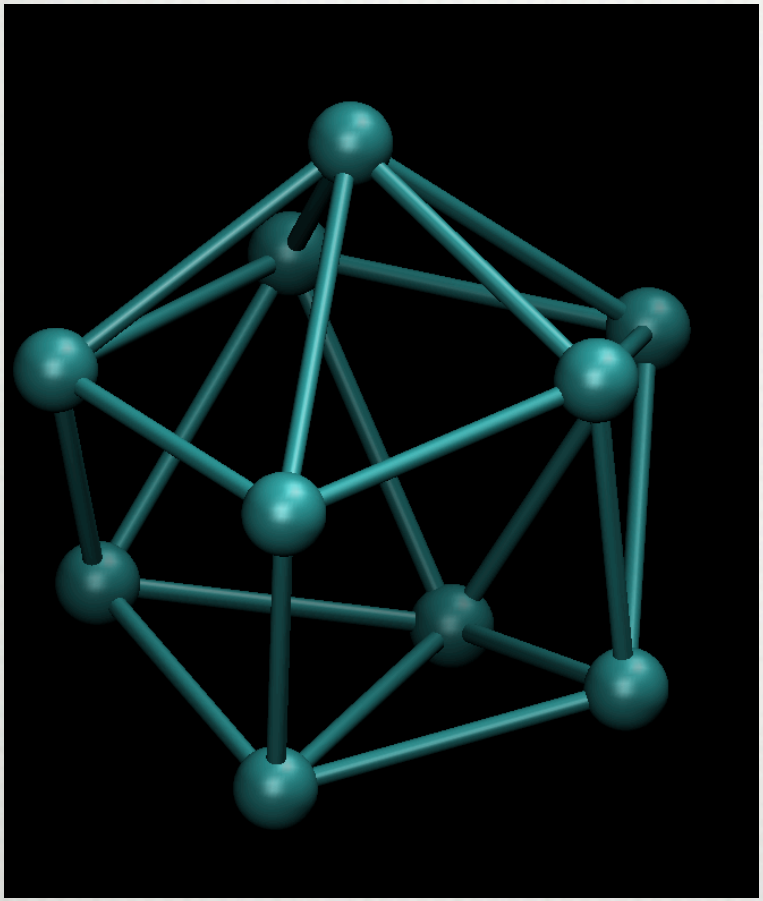
EXAMPLE: N=8

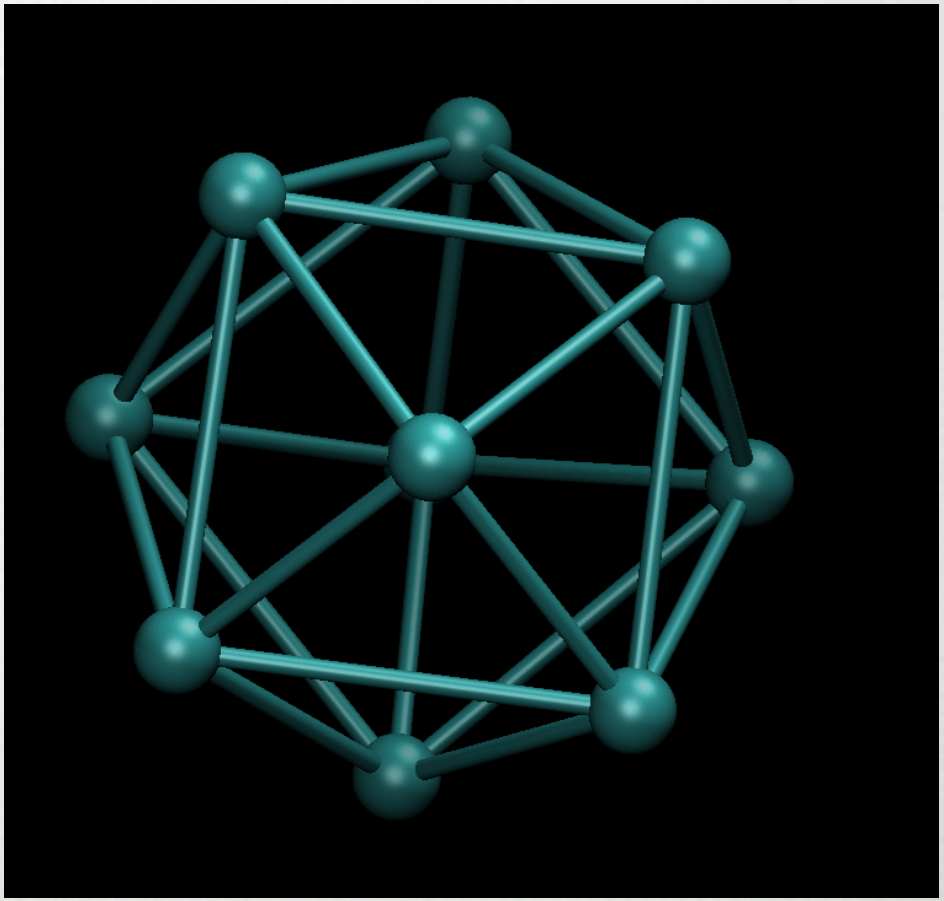
- NB=12
- NCROSS=20
- NMUT=5%
- nmut0=10
- sample size: 32
- Nv=16
- 20,000 generations

E=19.67685

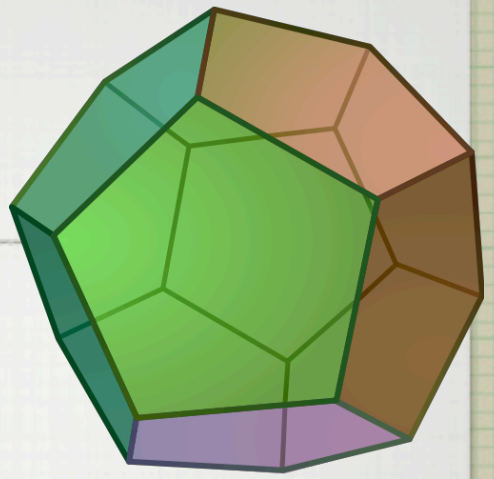


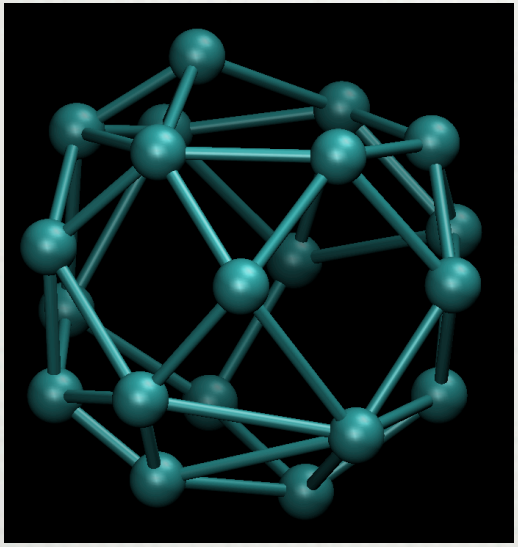
EXAMPLE N=20 (ISOCAHEDRON)



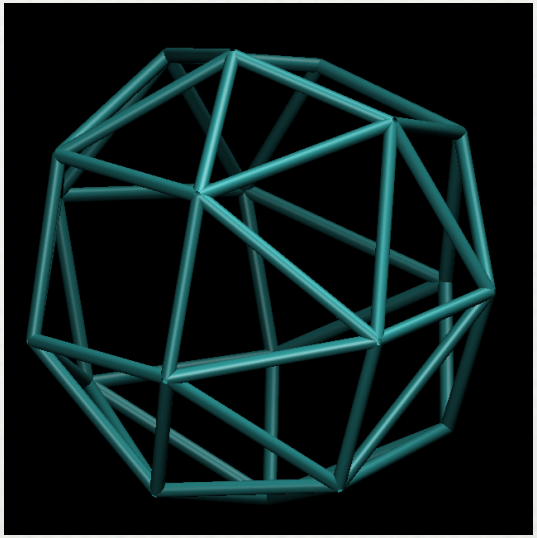


EXAMPLE N=40 (DODECAHEDRON)

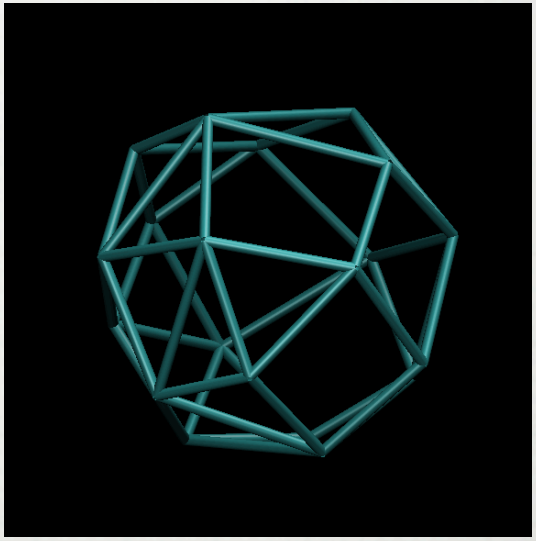




152.238



151.259



151.831

ADVANTAGES OF GA

- Concept is easy to understand
- Modular, separate from application
- Good for “noisy” environments
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed
- Many ways to speed up and improve a GA-based application as knowledge about problem domain is gained
- Easy to exploit previous or alternate solutions
- Substantial history and range of use

WHEN TO USE GA?

- ☐ **Alternate solutions are too slow or overly complicated**
- ☐ **Need an exploratory tool to examine new approaches**
- ☐ **Problem is similar to one that has already been successfully solved by using a GA**
- ☐ **Want to hybridize with an existing solution**
- ☐ **Benefits of the GA technology meet key problem requirements**

SUMMARY

- Directed search algorithms based on the mechanics of biological evolution
- Developed by John Holland, University of Michigan (1970's)
- 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
- Widely-used today in business, scientific and engineering circles

SUMMARY

- Genetic algorithms are—
- Fun! They are enjoyable to program and to work with
- This is probably why they are a subject of active research
- Mind-bogglingly slow—you don't want to use them if you have any alternatives
- Good for a very few types of problems
- Genetic algorithms can sometimes come up with a solution when you can see no other way of tackling the problem