Bachelor of Computer Science (Hons)
Year-2 Sep 2023

Welcome to Intelligent Systems

CAI3204

HIT THE GROUND RUNNING.™

Learning Objectives

- At the end of the course, students will be able to:
  - CO1: Identify the types of problem that are amenable to "intelligent" solutions.
  - CO2: Compare and contrast the various intelligent system techniques to solve such problems.
  - CO3: Select and apply appropriate intelligent techniques to a given problem.
  - CO4: Critically discuss intelligent system research issues and their applications.

Artificial Neural Networks

Learning Objectives

- ❑ At the end of the course, students will be able to:
  - ❑ CO1: Identify the types of problem that are amenable to "intelligent" solutions.
  - ❑ CO2: Compare and contrast the various intelligent system techniques to solve such problems.
  - ❑ CO3: Select and apply appropriate intelligent techniques to a given problem.
  - ❑ CO4: Critically discuss intelligent system research issues and their applications.

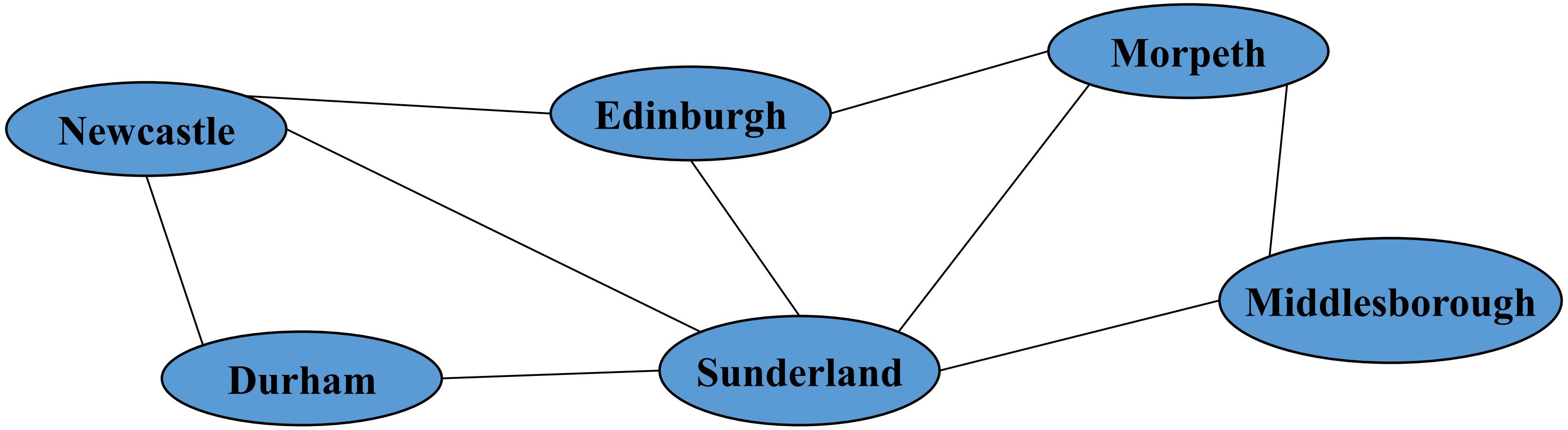
Genetic Algorithm

Genetic Algorithms

A way of solving difficult problems, inspired by natural selection ('survival of the fittest')

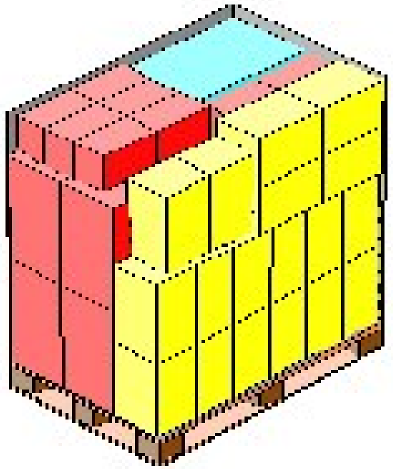
- What are they good for?
  - Example problems
  - Why are they difficult
- How do GAs work?
  - In theory (simple)
  - In practice (bit more tricky)

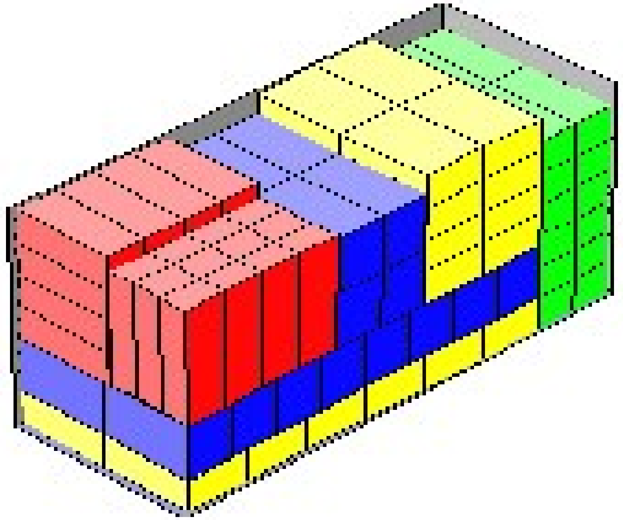
The Travelling Salesman



- Wants to visit each town (hard constraint), with minimum mileage (soft constraint)
- NP-Complete
- Similar to many networking problems

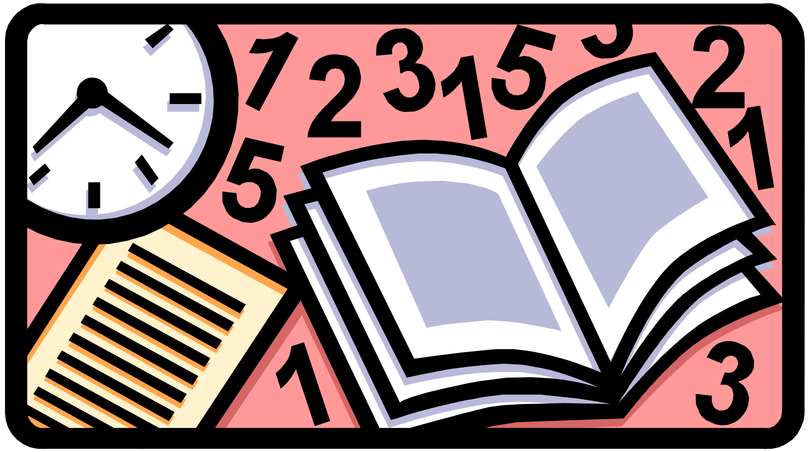
Knapsack Problems





- How to load X different containers with Y different packages
- Want to minimize wasted space using different combinations

Timetabling



- KDU has 10,000s students on 1000s of modules taught by 1000s staff in 100s of rooms of dozens of types
- Want to:
  - Minimise clashes (students, staff, rooms)
  - Maximise use of rooms

Scheduling

- Eg POS Laju
- Thousands of people in thousands of vans picking up and delivering millions of items
  - Most efficient schedules?

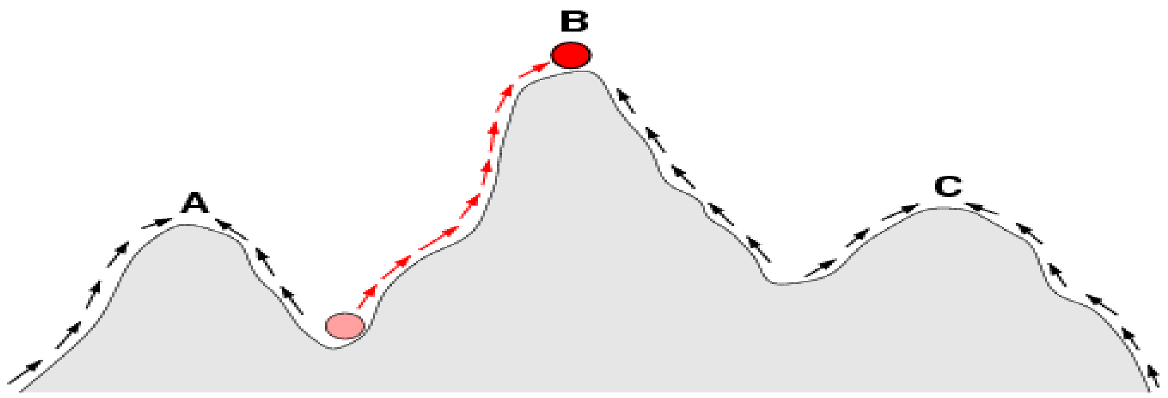
Parameter Optimisation

- Many systems rely on a lot of ‘free parameters’ that need fine tuning to get the best results
- Eg Agents in games may have
  - Tendency to fire
  - Likelihood of going toward enemy
  - Probability of make alliances
- What combination of parameter values are best?

Multivariate Optimisation Problems

- What do these problems (and many others) have in common?
  - Mutivariate optimisation – finding best result in a large 'space' of possibilities
  - For each possibility (ie each possible route, timetable, combination of blocks, set of parameter values) there is a 'score' of how good it is
  - Little or no decomposition into subproblems – changing one factor can effect every other one

Fitness Landscape



- Can picture the fitness of each possible solution as a landscape.
- The problem is then to find the highest peak (or lowest valley)
- The 'rougher' the landscape, the harder the problem

Other Multivariate Optimisation Strategies

- Brute force:
  - Try every possible combination to find the best
  - *I.e.* every possible timetable, combination of bricks, routes round a map, combination of parameter values, etc
  - Sometimes appropriate, often too computationally intensive
- Random search:
  - Try possibilities at random until you have one that's good enough
  - Sometimes good enough.

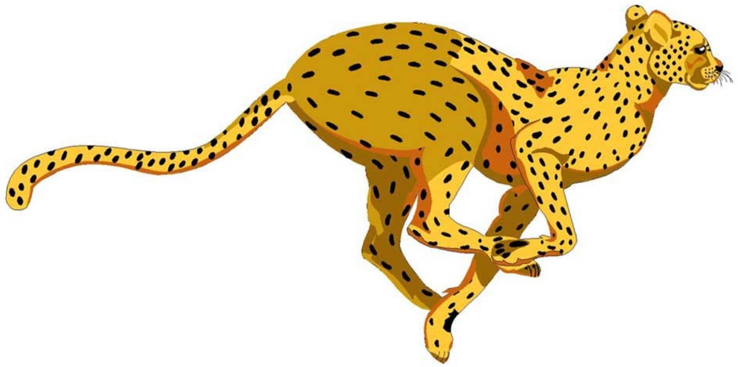
Heuristics

- In many applications there are simple ‘rules of thumb’ that produce pretty good answers
  - Travelling salesman: start at one town, then go to the nearest one you haven’t visited yet
  - Knapsack: put the biggest item in that will fit, and repeat
- But in some cases can produce very poor answers

Genetic Algorithms

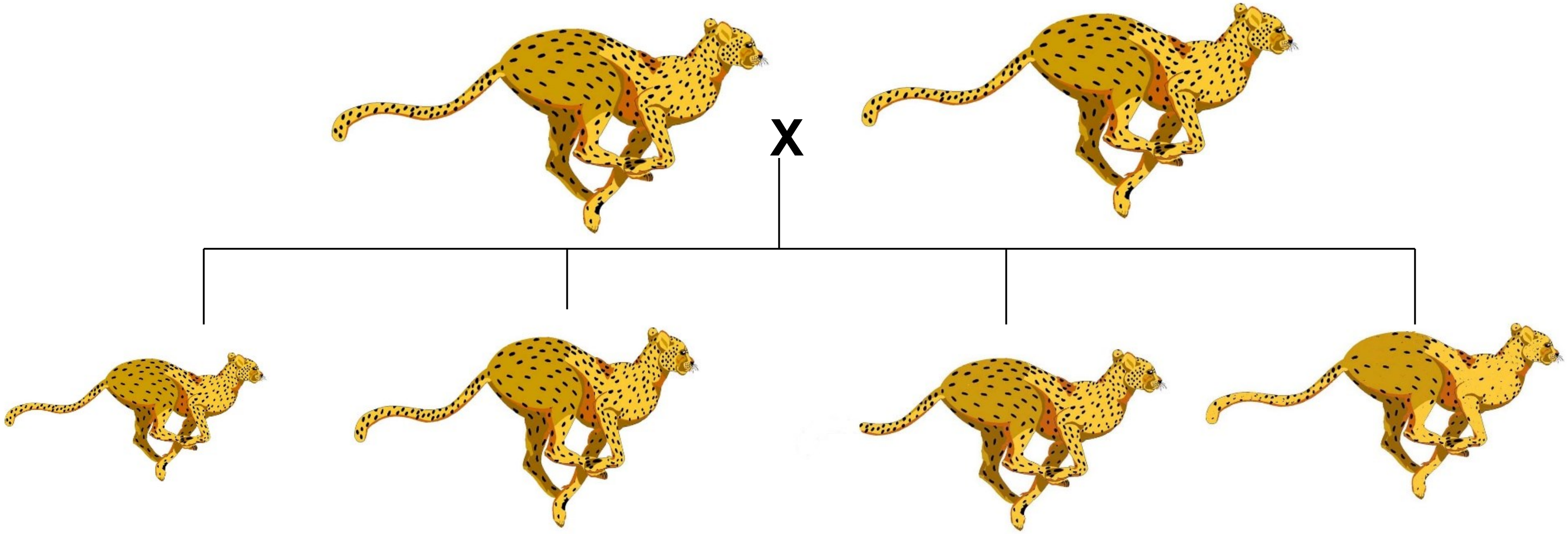
- A technique for solving multivariate optimisation problems
  - (relatively) simple
  - General purpose
  - (fairly) efficient
  - (fairly) reliable
- Inspired by natural selection ('survival of the fittest')

Natural Selection



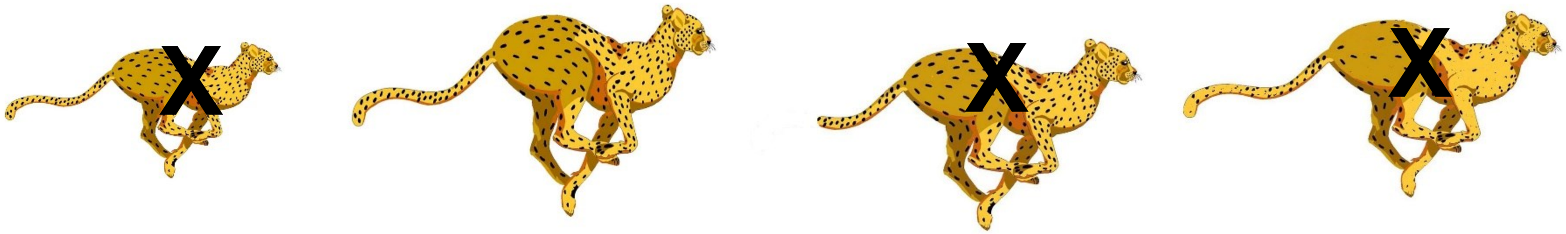
- Question: Why are cheetahs fast (or trees tall, or humans intelligent, or MRSA infectious)?
- Answer: A combination of two processes:
  - *Descent with modification*
  - *Survival of the fittest*

Descent with Modification



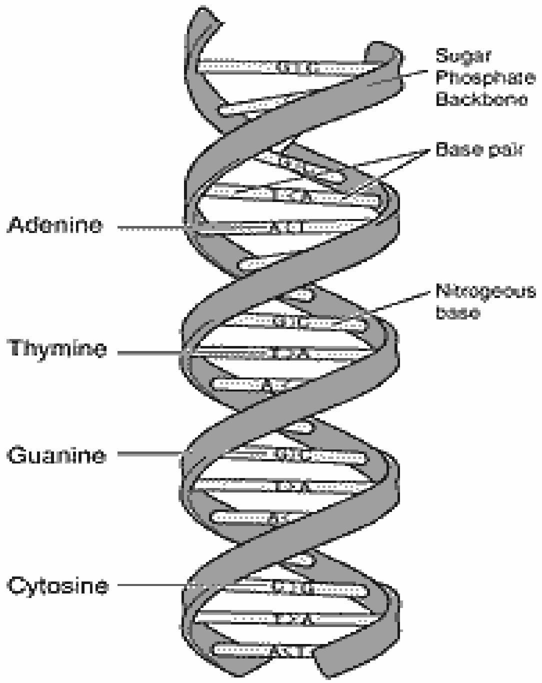
- When cheetahs breed, the offspring are similar, *but not identical to*, the parents

Survival of the Fittest

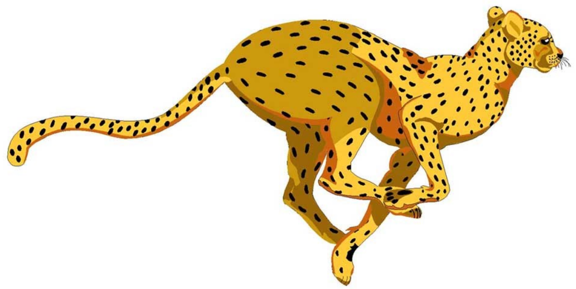


- The offspring will be more or less successful at surviving and breeding, dependent on their inherited characteristics
- Over many generations cheetahs become better *adapted* to their environment

Genotypes and Phenotypes



... C A G G T A C G A A A ...



- Genes encoded in base-pairs in DNA chromosomes
- *Genotype* = the gene sequence of an individual
- *Phenotype* = the body generated from the genes

Reproduction

- How are cheetahs reproduced at the genetic level?
  - Recombination
  - Mutation
- Recombination ensures that genes from parents are carried over into new generation (*descent*)
- Mutation ensures that new possibilities are created (*modification*)

Recombination

C A G G T A C G A A   X   T G A C T G C A A A

C A G G T G C A A A

- Combine genes from both parents
- Crossover at random location(s)
- Result is similar to, but not identical to both parents

Mutation

C A G G T G C A A A

↓

C A G G T G C T A A

- Make random point changes at individual genes
- *Mutation rate* = chance that any one gene will be changed

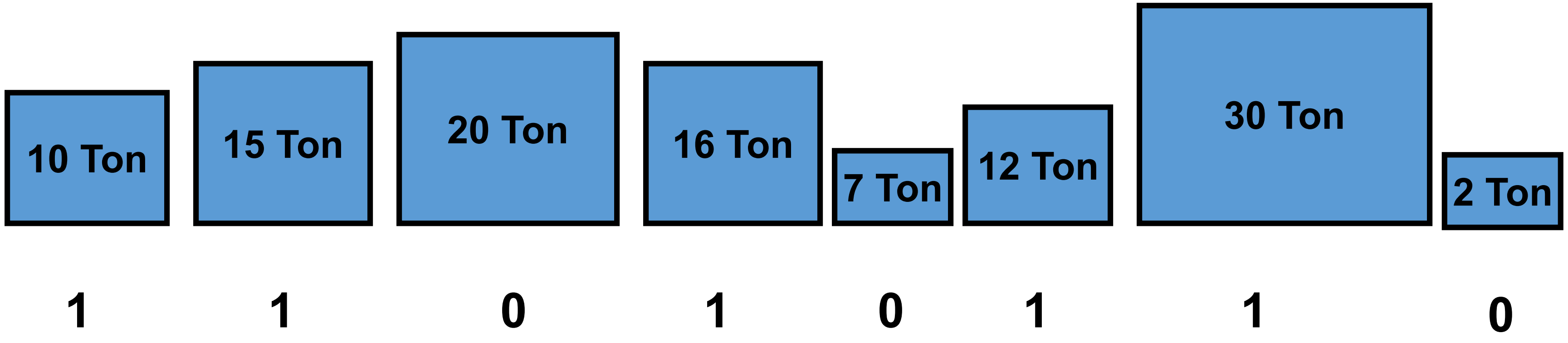
Genetic Algorithm

1. Find a *representation*, a way of encoding each possible solution onto a string of symbols
2. Generate a *population* of random genomes
3. Find the *fitness* of each member of the population
4. ‘Breed’ the next *generation* of individuals
5. Repeat 3,4 until you find a *satisfactory* solution

Genetic Representation

- A way of representing the possible solutions as a string of symbols
  - Typically 1s and 0s
- Each combination can be represented
- Each combination represented just once

Genetic Representation

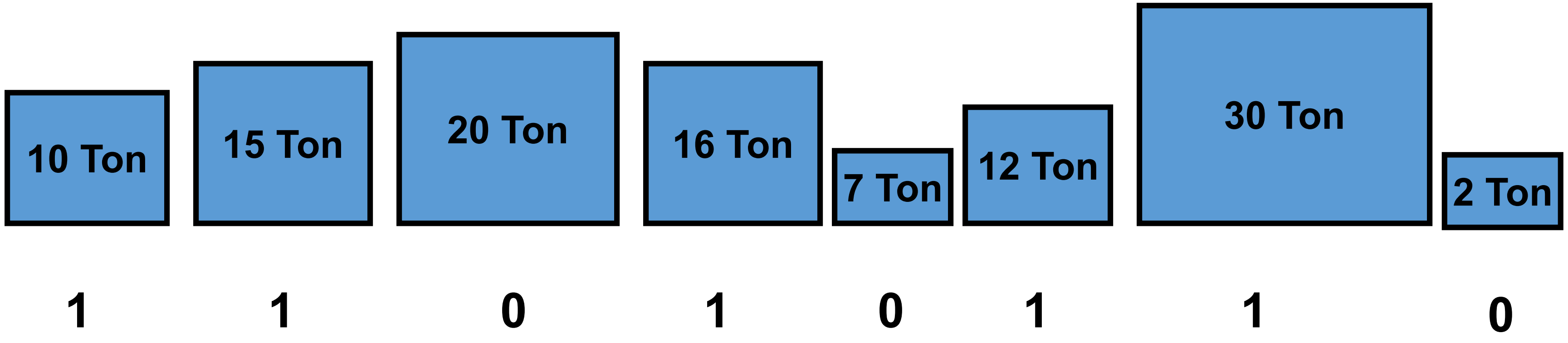


- *Eg* Knapsack Problem: *Truck should take 44 ton. What load combination is best?*
- 128 possible combinations (*ie* a small problem)
- Each one represented as 7-bit string

Fitness Function

- A score of how good each possible solution is
  - Better = lower
  - *Or* better = higher

Fitness Function



- *Eg* Fitness(11010110) = | 83 – 44 | = 39
- Lower = Better in this case

Populations and Generations

- Start by producing an *initial population* of random individuals (generation 0)

<u>Generation 0</u>
11010110
01010101
11011001
01011010
01001011
11001010

Find the fittest

- Find the fitness of each individual

<u>Generation 0</u>
F(11010110)=39
F(01010101)=1
F(11011001)=5
F(01011010)=24
F(01001011)=10
F(11001010)=18

Survival of the fittest

- Find the fittest individuals

<u>Generation 0</u>

F(11010110)=39

F(01010101)=1

F(11011001)=5

F(01011010)=24

F(01001011)=10

F(11001010)=18

Descent with Modification

- And ‘breed’ using recombination

Generation 0		Generation 1
11010110		01011001
01010101	x	11010101
11011001		01011001
01011010	x	11011011
01001011		01001001
11001010		11001011

Descent with Modification

- Mutate the new population

<u>Generation 1</u>

01011001
11011101
01011001
11011011
01001001
10001011

Next Generation

- And find the fitness of the new generation
- Repeat until a suitable solution is found

<u>Generation 1</u>

F(01011001)=4

F(11011101)=18

F(01011001)=4

F(11011011)=32

F(01001001)=20

F(10001011)=5

Elitism (Superiority)

- **Problem**: on rough fitness landscapes, F(AxB) >> F(A),F(B) is possible
- *ie* Recombination may *not* result in reliably good fitness. May lose good solutions
- So, ‘copy over’ some proportion of the population from the old to the new generation.

Elitism

Generation 0

F(11010110)=39

F(01010101)=1

F(11011001)=5

F(01011010)=24

F(01001011)=10

F(11001010)=18

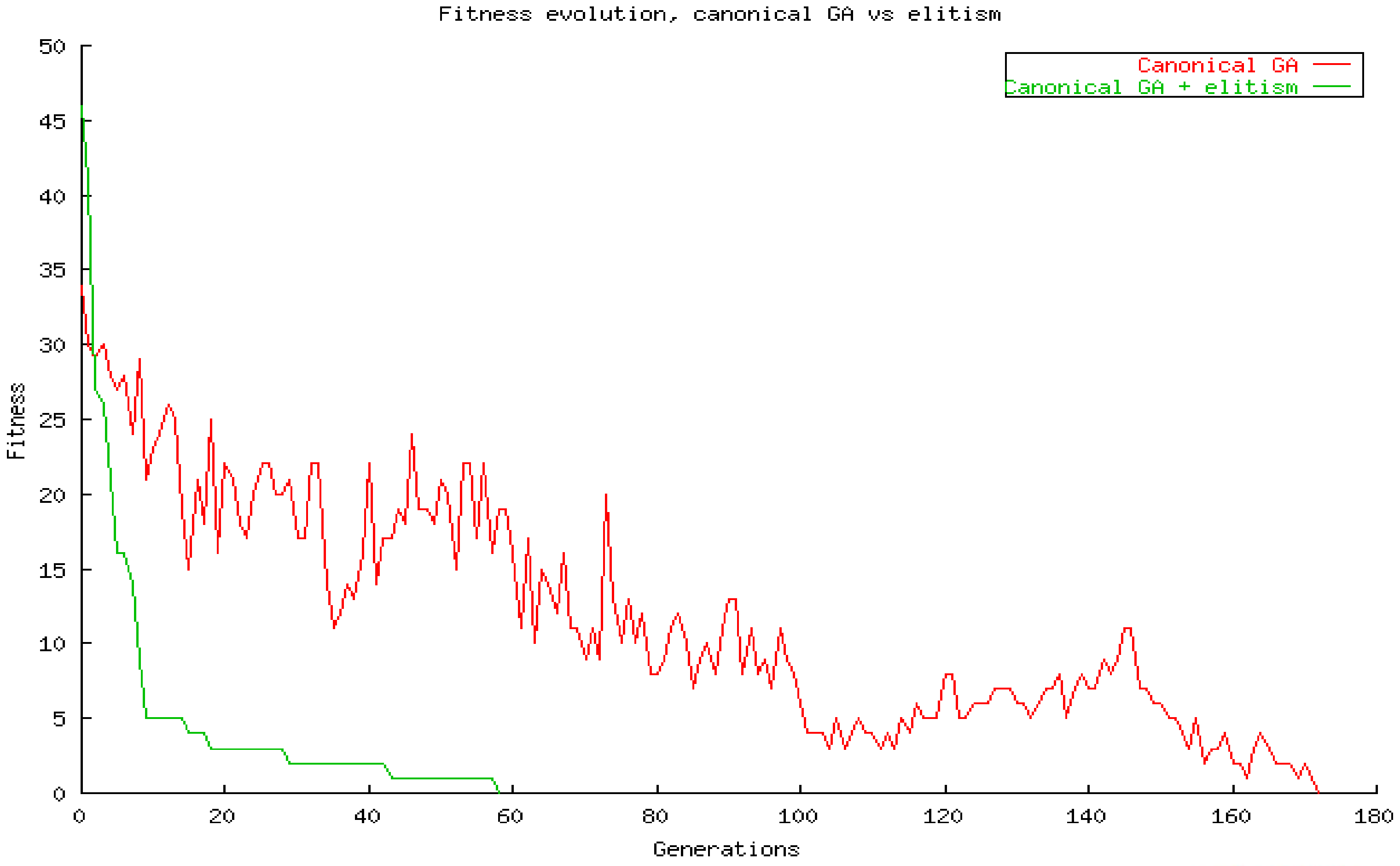
Generation 1

01010101

11011001

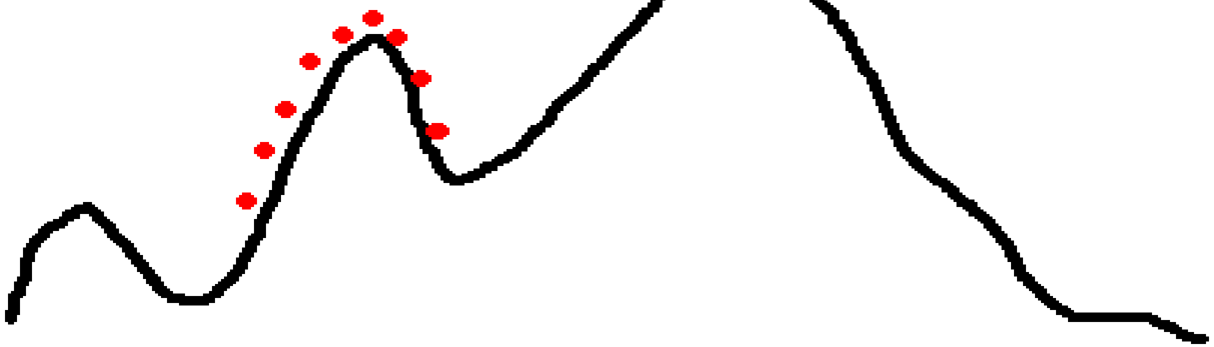
…

Evolution with/out Elitism



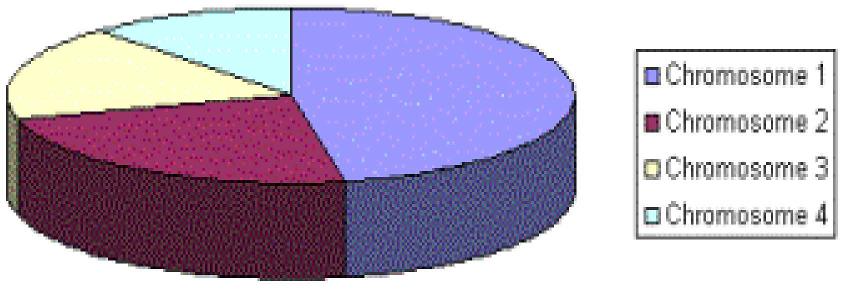
Premature Convergence

- **Problem**: too much elitism can make the population 'converge' prematurely
- (Similar to inbreeding in wild populations)
- Population can stick on a local optima



Roulette Wheel Selection

- So select parents of next generation from throughout population using a ‘roulette wheel’
- Chances of any individual being chosen are (inversely) proportional to their fitness
- Many more individuals have a chance of contributing the next generation
- Like balls landing in a rigged roulette wheel
- Used in conjunction with elitism



Tournament Selection

- Roulette wheel turns out to be awkward to calculate
- Tournament selection is nearly as effective:
  1. Randomly select *A* and *B* from population
  2. If *fitness(A)>fitness(B)* then breed from *A*.
  3. Else, breed from *B*
- Given a large enough population, then chances of being selected are proportional to fitness

Competency

- GA are not guaranteed to work faster than random search
- Computationally expensive:
  - Eg 100 population x 1000 generations = 100,000 evaluations of fitness function
- A lot of current work is in finding ‘competent’ GA’s – ie ways of guaranteeing amount of computation and goodness of solution

Summary

- GA s are a simple but powerful technique
- Not always guaranteed to work
- Huge amount of material available:
  - Holland *Genetic Algorithms*, Scientific American
  - Goldberg, David E (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*
  - Mitchell, Melanie, (1996), *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA Addison-Wesley
  - http://en.wikipedia.org/wiki/Genetic_algorithm

Demo- PasswordGASeq

Adapted Slide (Extra for you to consider)

Genetic Algorithm for Variable Selection

Jennifer Pittman
ISDS
Duke University

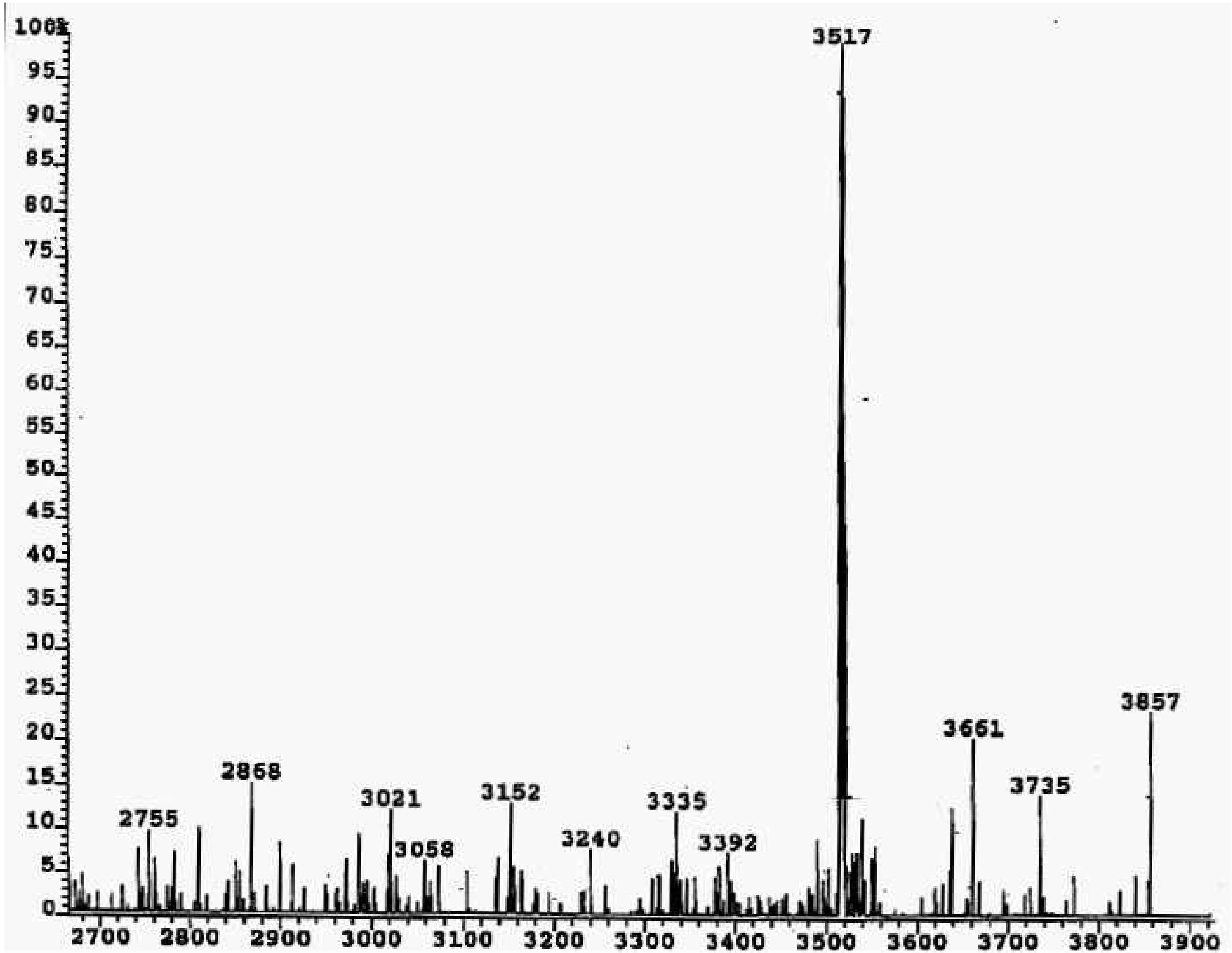
Genetic Algorithms Step by Step

Jennifer Pittman

ISDS

Duke University

Example: Protein Signature Selection in Mass Spectrometry



relative intensity

molecular weight

http://www.uni-mainz.de/~frosc000/fbg_po3.html

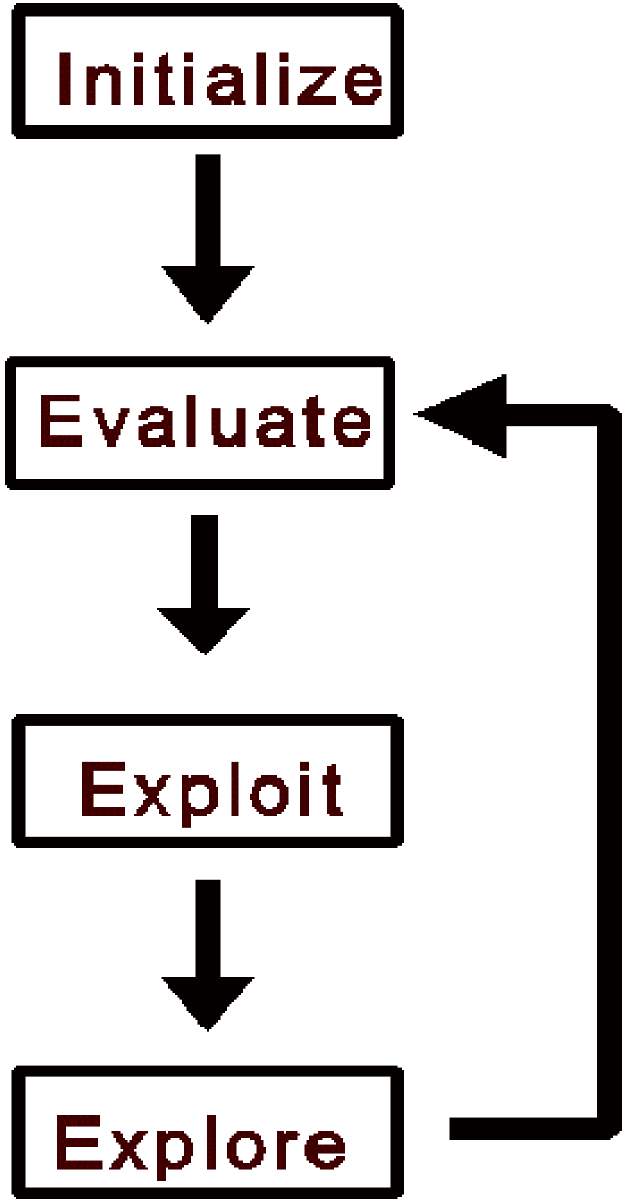
Genetic Algorithm (Holland)

- heuristic method based on ' survival of the fittest '
- useful when search space very large or too complex for analytic treatment
- in each iteration (generation) possible solutions or individuals represented as strings of numbers

3021 3058 3240

00010101 00111010 11110000
00010001 00111011 10100101
00100100 10111001 01111000

11000101 01011000 01101010

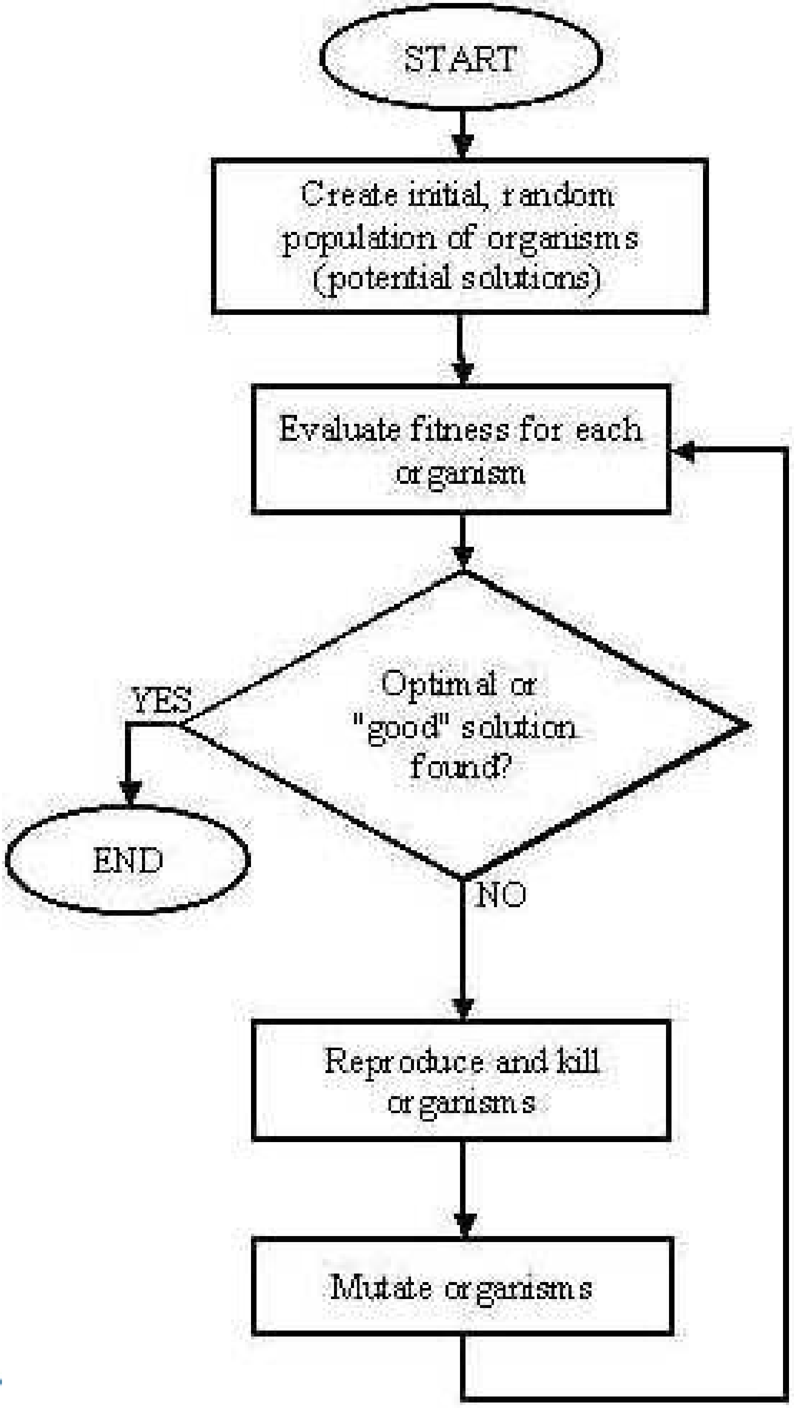


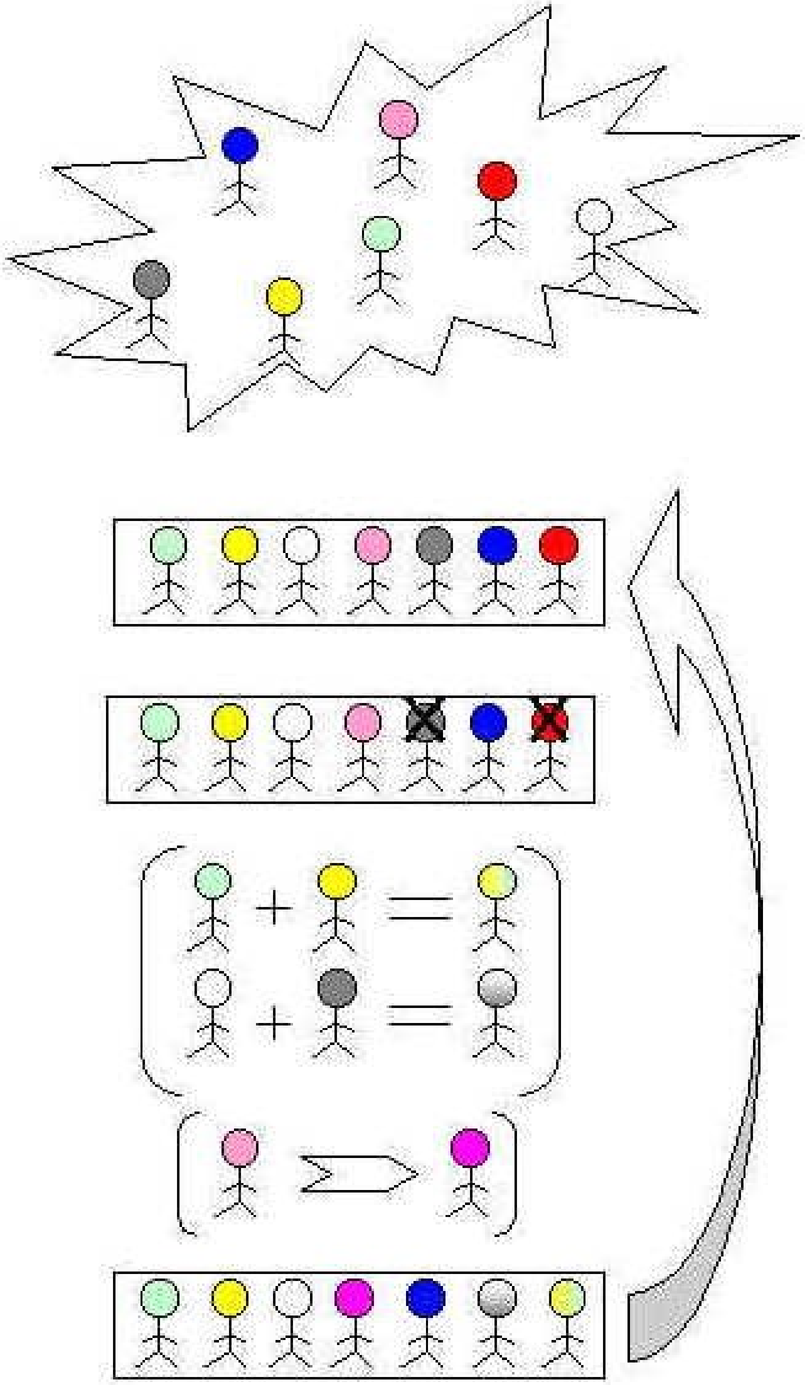
- all individuals in population evaluated by fitness function

- individuals allowed to reproduce (selection), crossover, mutate

Flowchart of GA

http://ib-poland.virtualave.net/ee/genetic1/3geneticalgorithms.htm





(a simplified example)

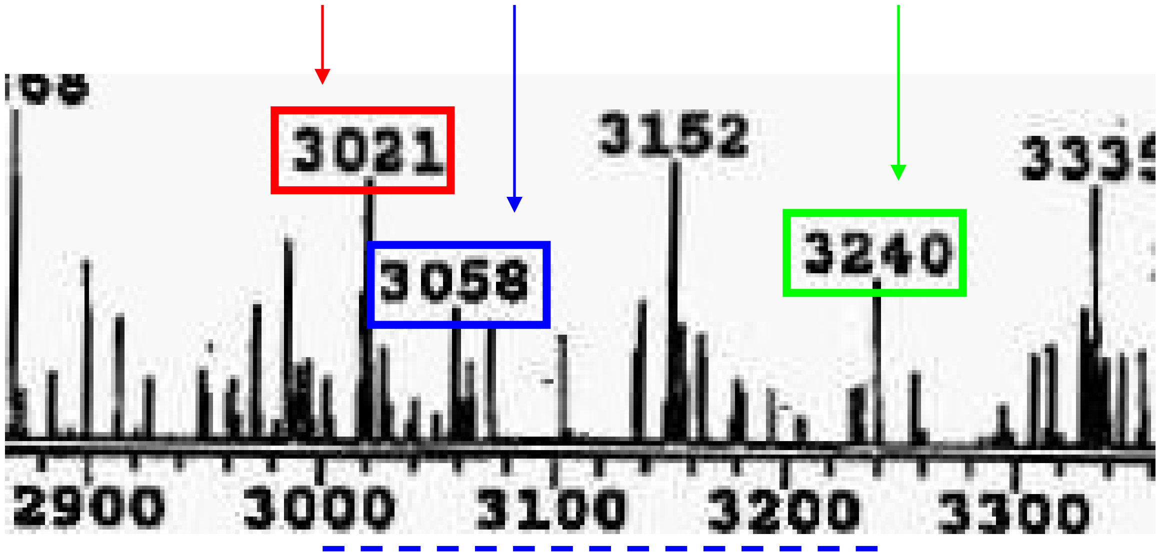
Initialization

- proteins corresponding to 256 mass spectrometry values from 3000-3255 m/z

- assume optimal signature contains 3 peptides represented by their m/z values in binary encoding

- population size ~M=L/2 where L is signature length

00010101 00111010 11110000



Initial Population

00010101 00111010 11110000
00010001 00111011 10100101
00100100 10111001 01111000

M = 12

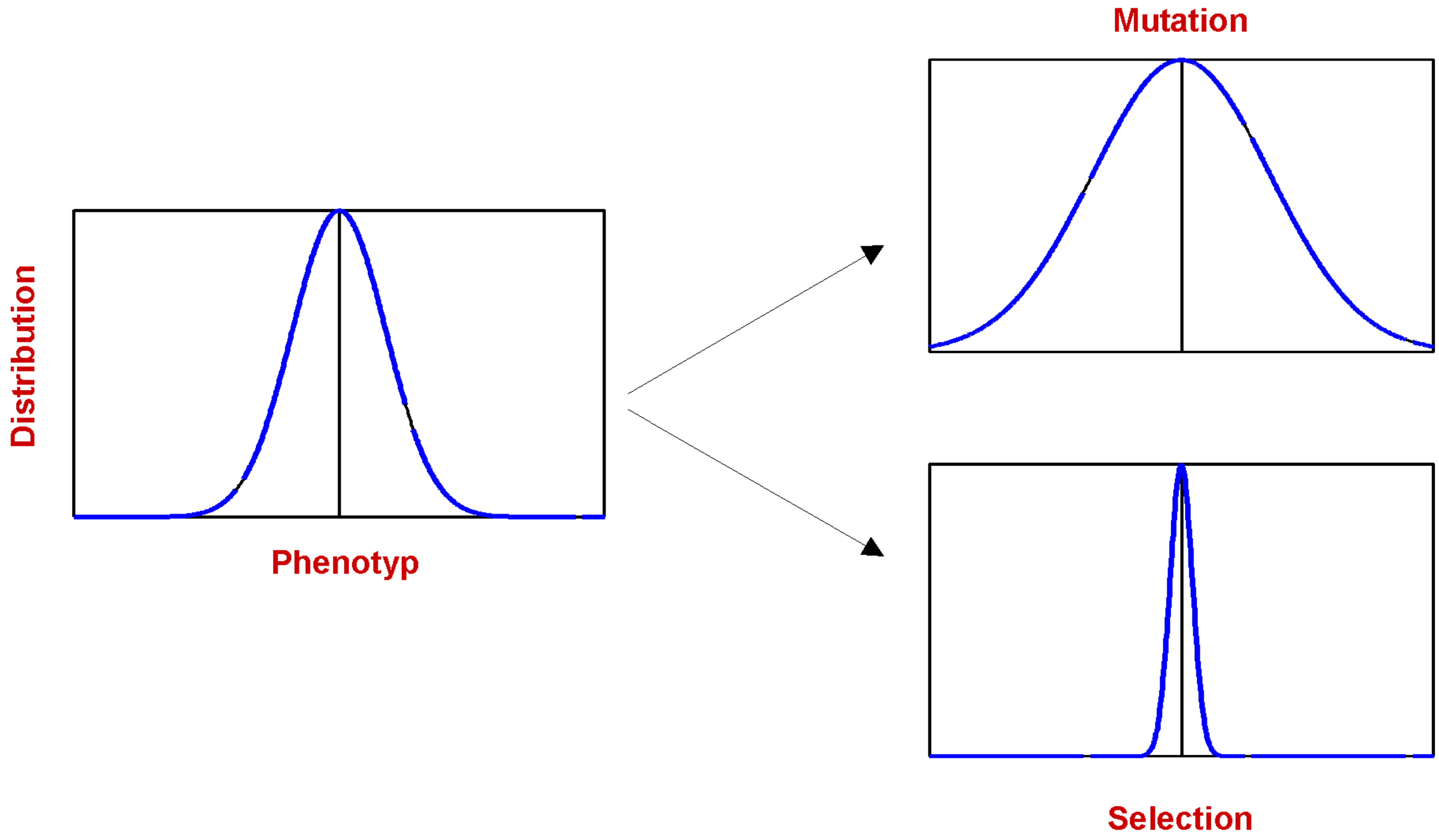
11000101 01011000 01101010

L = 24

Searching

- search space defined by all possible encodings of solutions
- selection, crossover, and mutation perform ‘pseudo-random’ walk through search space
- operations are non-deterministic yet directed

Phenotype Distribution



http://www.ifs.tuwien.ac.at/~aschatt/info/ga/genetic.html

Evaluation and Selection

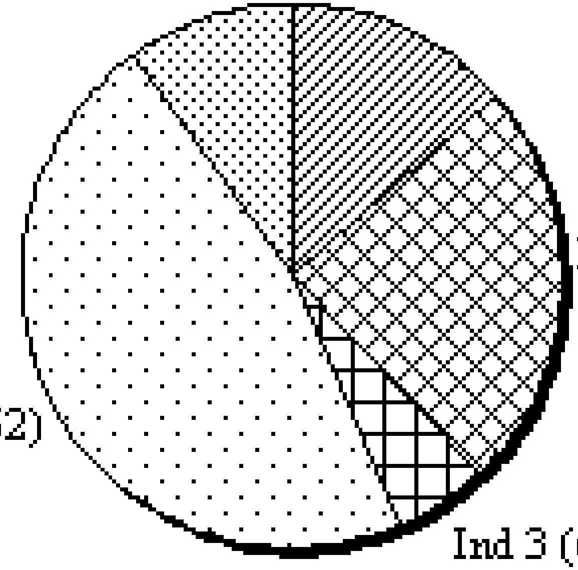
- evaluate fitness of each solution in current population (e.g., ability to classify/discriminate) [involves genotype-phenotype decoding]

- selection of individuals for survival based on probabilistic function of fitness

- on average mean fitness of individuals increases

- may include elitist step to ensure survival of fittest individual

Population	*Fitness*
Individual 1	15
Individual 2	27
Individual 3	6
Individual 4	52
Individual 5	11



Randomly generated number = 21

Individual 2 is selected

Roulette Wheel Selection

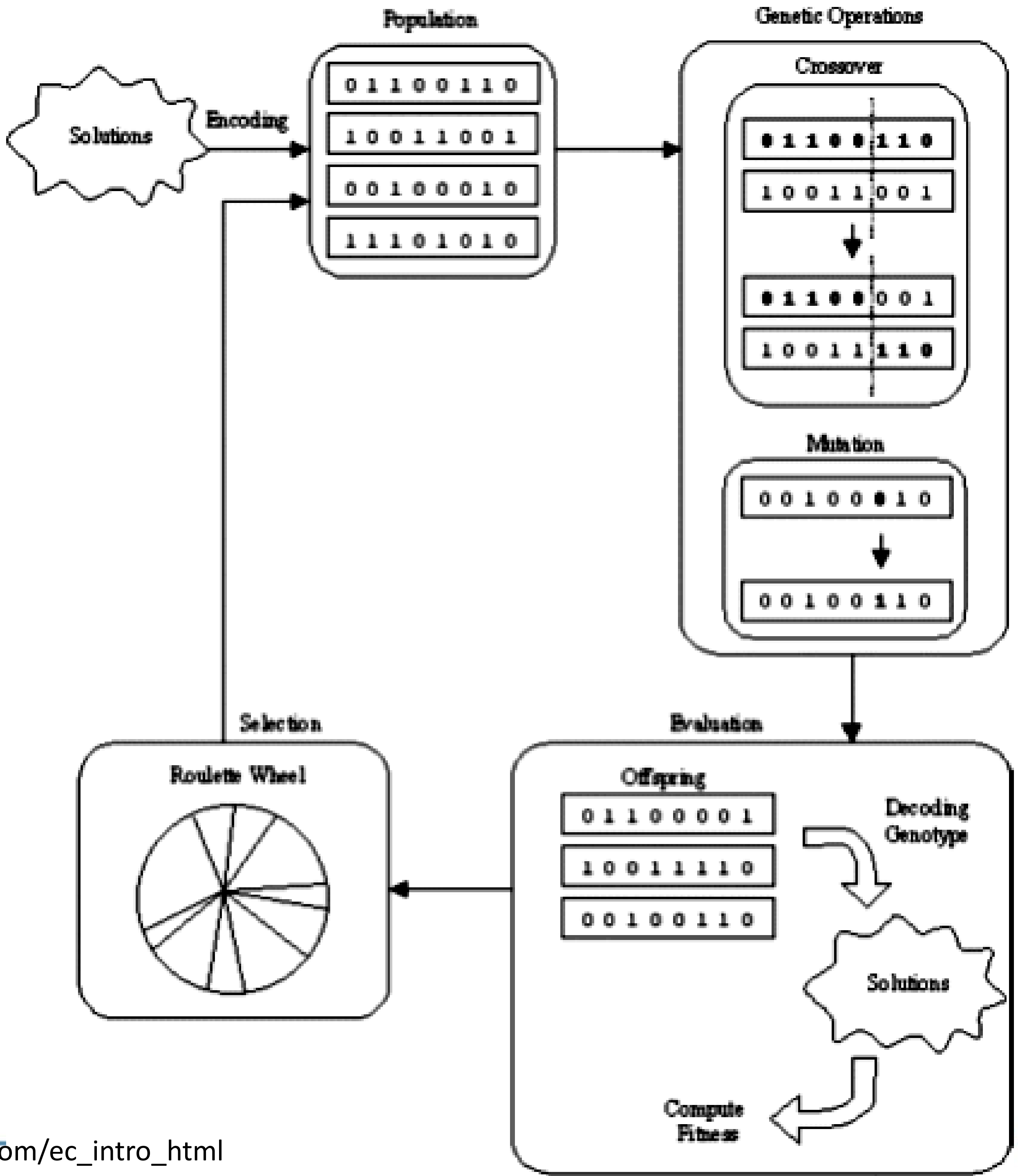
Crossover

- combine two individuals to create new individuals for possible inclusion in next generation
- main operator for local search (looking close to existing solutions)
- perform each crossover with probability p_c {0.5,…,0.8}
- crossover points selected at random
- individuals not crossed carried over in population

Initial Strings		Offspring
Single-Point		
11000101 01011000 01101010	→	11000101 01011001 01111000
00100100 10111001 01111000	→	00100100 10111000 01101010
Two-Point		
11000101 01011000 01101010	→	11000101 01111001 01101010
00100100 10111001 01111000	→	00100100 10011000 01111000
Uniform		
11000101 01011000 01101010	→	01000101 01111000 01111010
00100100 10111001 01111000	→	10100100 10011001 01101000

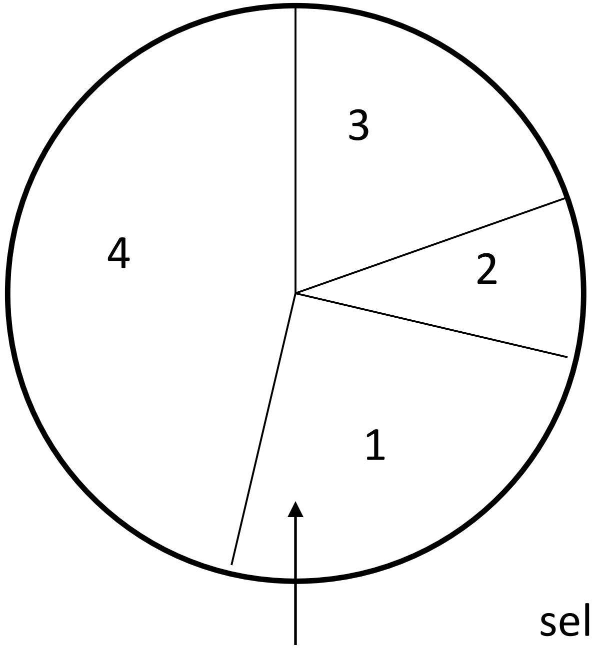
Mutation

- each component of every individual is modified with probability p_m
- main operator for global search (looking at new areas of the search space)
- p_m usually small {0.001,…,0.01}
  rule of thumb = 1/no. of bits in chromosome
- individuals not mutated carried over in population



Population
Genetic Operations
Crossover
Solutions
Encoding
0 1 1 0 0 1 1 0
1 0 0 1 1 0 0 1
0 0 1 0 0 0 1 0
1 1 1 0 1 0 1 0
Mutation
Selection
Evaluation
Roulette Wheel
Offspring
0 1 1 0 0 0 0 1
1 0 0 1 1 1 1 0
0 0 1 0 0 1 1 0
Decoding Genotype
Solutions
Compute Fitness

phenotype		genotype	fitness
3021 3058 3240		00010101 00111010 11110000	0.67
3017 3059 3165	→	00010001 00111011 10100101	0.23
3036 3185 3120		00100100 10111001 01111000	0.45
3197 3088 3106		11000101 01011000 01101010	0.94



selection

1		00010101 00111010 11110000
3	→	00100100 10111001 01111000
4		11000101 01011000 01101010
4		11000101 01011000 01101010

one-point crossover (p=0.6)

	00010101 00111010 11110000		00010101 00111001 01111000
0.3	00100100 10111001 01111000	→	00100100 10111010 11110000
	11000101 01011000 01101010		11000101 01011000 01101010
0.8	11000101 01011000 01101010		11000101 01011000 01101010

mutation (p=0.05)

	00010101 00111001 01111000		00010101 00110001 01111010
→	00100100 10111010 11110000	→	10100110 10111000 11110000
	11000101 01011000 01101010		11000101 01111000 01101010
	11000101 01011000 01101010		11010101 01011000 00101010

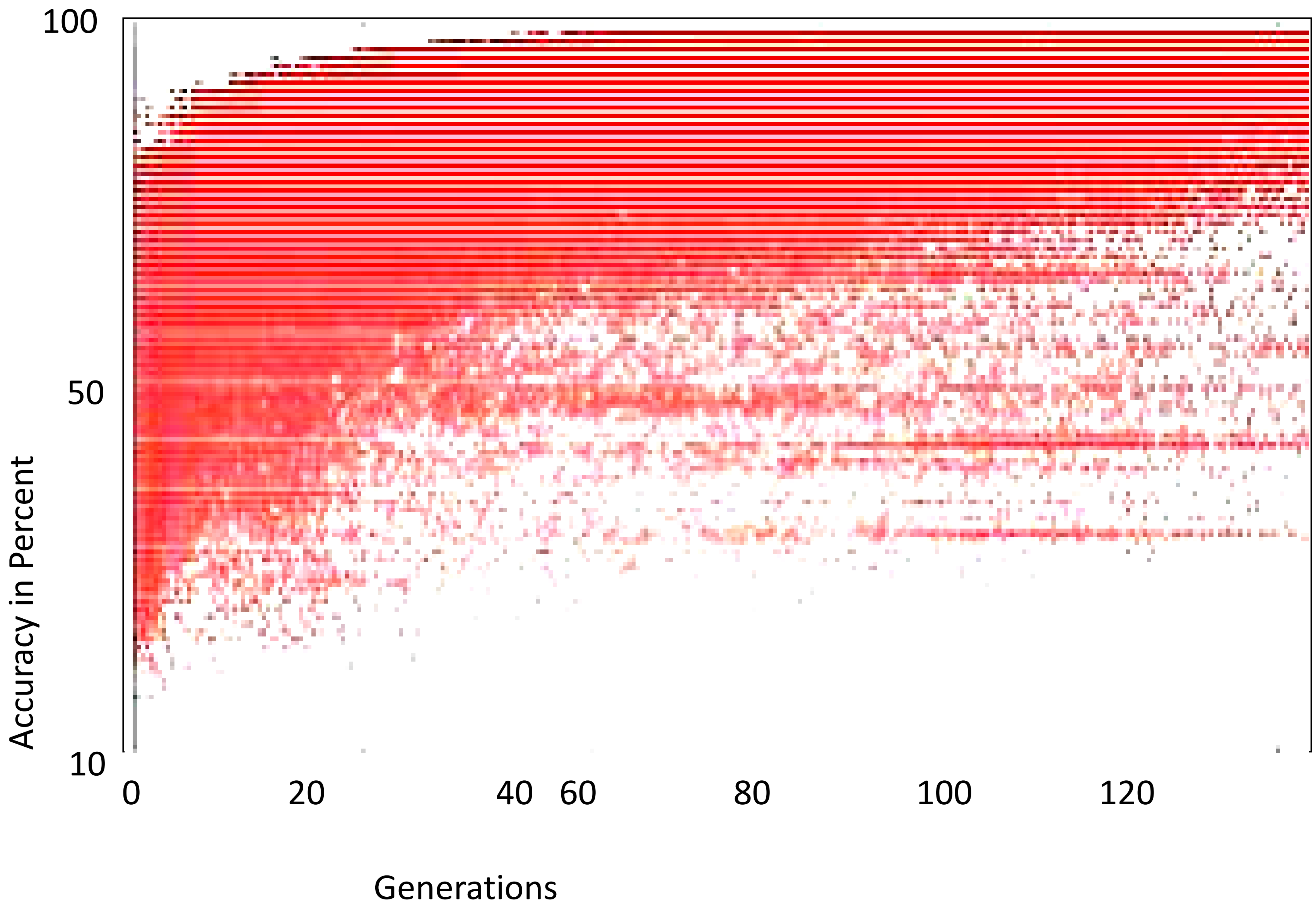
starting generation

3021 3058 3240	00010101 00111010 11110000	0.67
3017 3059 3165	00010001 00111011 10100101	0.23
3036 3185 3120	00100100 10111001 01111000	0.45
3197 3088 3106	11000101 01011000 01101010	0.94

next generation

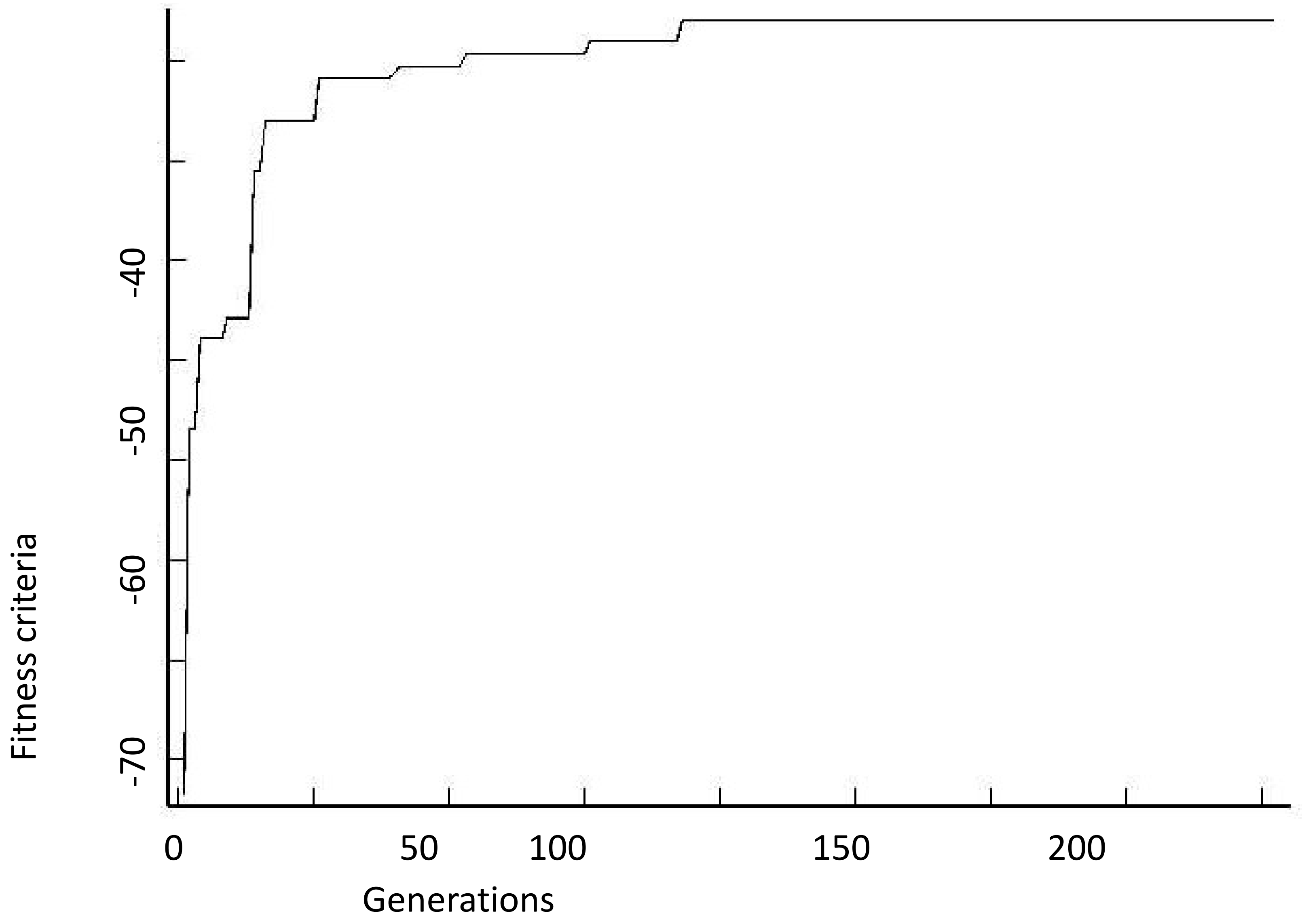
genotype		phenotype	fitness
00010101 00110001 01111010		3021 3049 3122	0.81
10100110 10111000 11110000	→	3166 3184 3240	0.77
11000101 01111000 01101010		3197 3120 3106	0.42
11010101 01011000 00101010		3213 3088 3042	0.98

GA Evolution

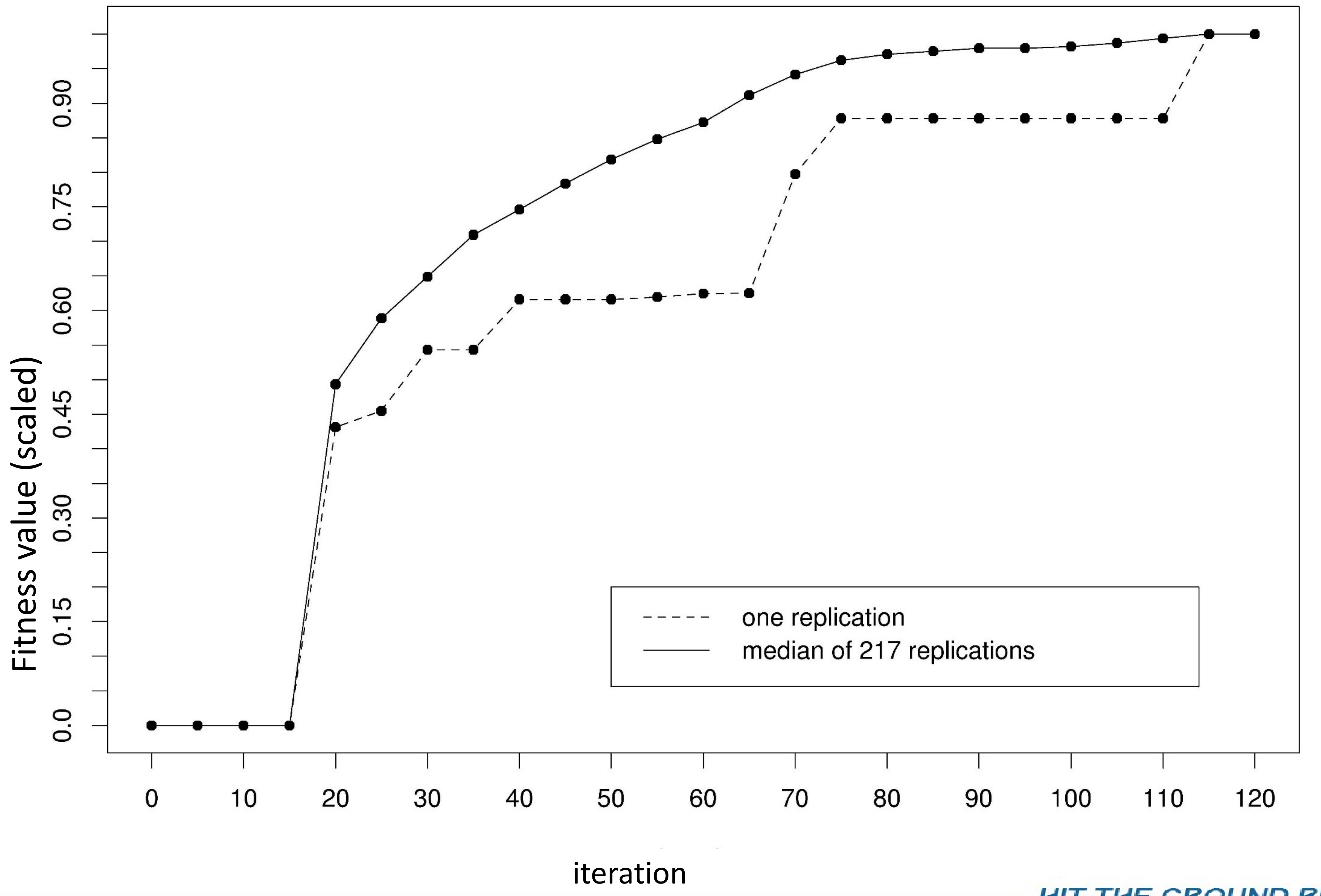


http://www.sdsc.edu/skidl/projects/bio-SKIDL/

genetic algorithm learning



http://www.demon.co.uk/apl385/apl96/skom.htm



Fitness value (scaled)
0.0
0.15
0.30
0.45
0.60
0.75
0.90
0 10 20 30 40 50 60 70 80 90 100 110 120
iteration
one replication
median of 217 replications

References

- Holland, J. (1992), *Adaptation in natural and artificial systems , 2nd Ed*. Cambridge: MIT Press.

- Davis, L. (Ed.) (1991), *Handbook of genetic algorithms.* New York: Van Nostrand Reinhold.

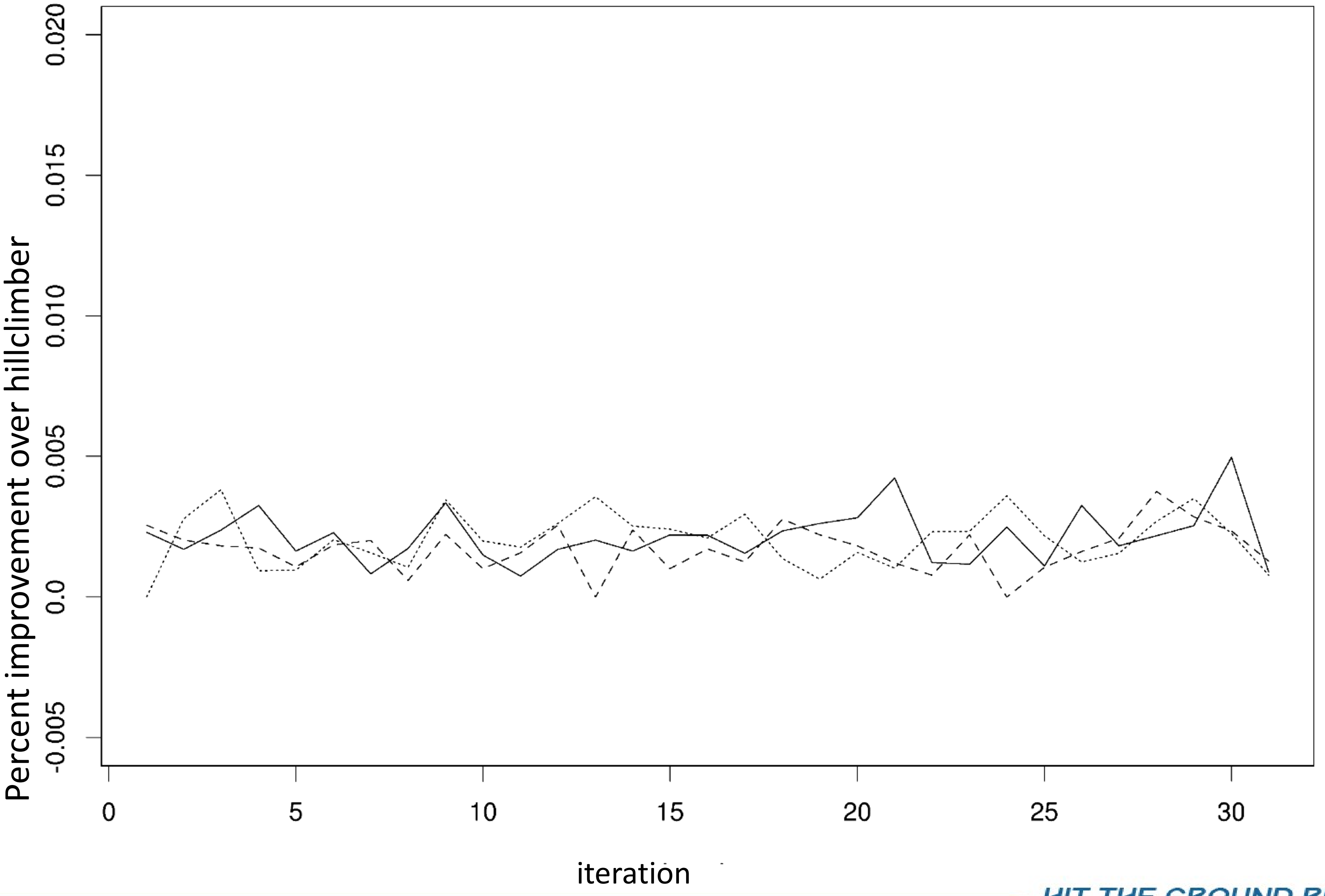
- Goldberg, D. (1989), *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley.

- Fogel, D. (1995), *Evolutionary computation: Towards a new philosophy of machine intelligence.* Piscataway: IEEE Press.

- Bäck, T., Hammel, U., and Schwefel, H. (1997), 'Evolutionary computation: Comments on the history and the current state', IEEE Trans. On Evol. Comp. 1, (1)

Online Resources

- http://www.spectroscopynow.com
- http://www.cs.bris.ac.uk/~colin/evollect1/evollect0/index.htm
- IlliGAL (http://www-illigal.ge.uiuc.edu/index.php3)
- GAlib (http://lancet.mit.edu/ga/)



Percent improvement over hillclimber
-0.005
0.0
0.005
0.010
0.015
0.020
0
5
10
15
20
25
30
iteration

Schema and GAs

- a schema is template representing set of bit strings

1**100*1 $\longrightarrow$ { 10010011, 11010001, 10110001, 11110011, … }

- every schema s has an estimated average fitness f(s):

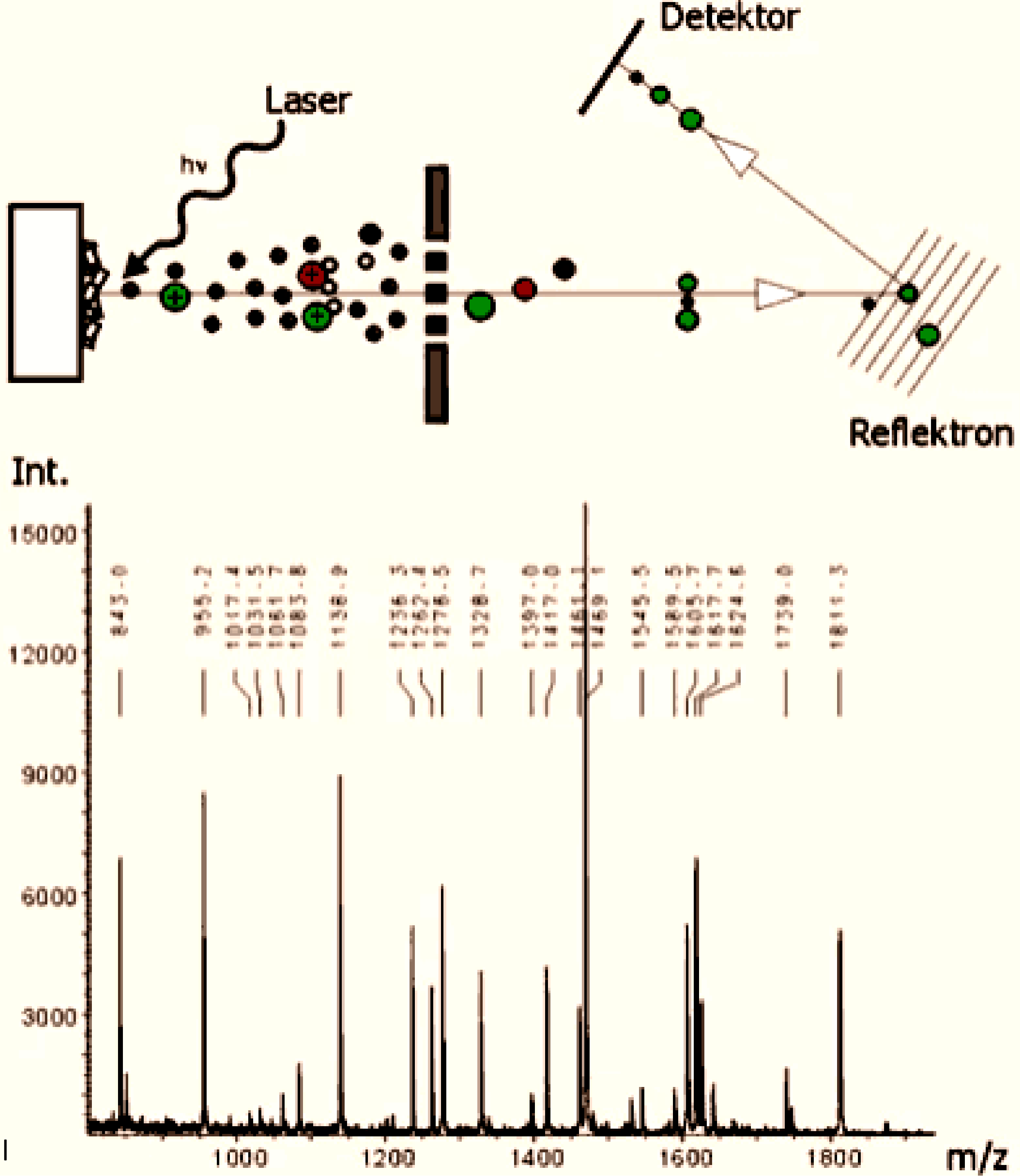
$$E_{t+1} \geq k \cdot [f(s)/f(pop)] \cdot E_t$$

- schema s receives exponentially increasing or decreasing numbers depending upon ratio f(s)/f(pop)

- above average schemas tend to spread through population while below average schema disappear

(simultaneously for all schema – ‘implicit parallelism’)

MALDI-TOF



©www.protagen.de/pics/main/maldi2.html

'UNNING.™

Stretch Break!

Seminar:

Thank you