Multidisciplinary System Design Optimization

A Basic Introduction to Genetic Algorithms

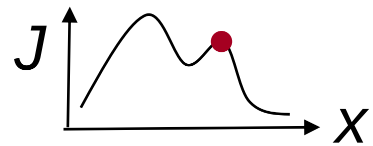
Lecture 11

Prof. Olivier de Weck

Heuristic Search Techniques

Main Motivation for Heuristic Techniques:

(1) To deal with local optima and not get trapped in them



(2) To allow optimization for systems, where the design variables are not only continuous, but discrete (categorical), integer or even Boolean

$x_i \notin \mathrm{IR}$ $\quad$ x_i ={1,2,3,4,5}, x_i ={'A','B','C'} x_i ={true, false}

These techniques do not guarantee that global optimum can be found. Generally Karush-Kuhn-Tucker conditions do not apply.

Principal Heuristic Algorithms

- Genetic Algorithms (Holland – 1975)
  – Inspired by genetics and natural selection – max fitness
- Simulated Annealing (Kirkpatrick – 1983)
  – Inspired by statistical mechanics– min energy
- Particle Swarm Optimization (Eberhart Kennedy - 1995)
  – Inspired by the social behavior of swarms of insects or flocks of birds – max "food"

⇒ These techniques all use a combination of <u>randomness</u> and heuristic "rules" to guide the search for global maxima or minima

Today: Genetic Algorithms

- Genetics and Natural Selection
- A simple genetic algorithm (SGA)
- “The Genetic Algorithm Game”
- Encoding - Decoding (Representation)
- Fitness Function - Selection
- Crossover – Insertion - Termination

Premise of GAs

- Natural Selection is a very successful organizing principle for optimizing individuals and populations of individuals

- If we can mimic natural selection, then we will be able to optimize more successfully

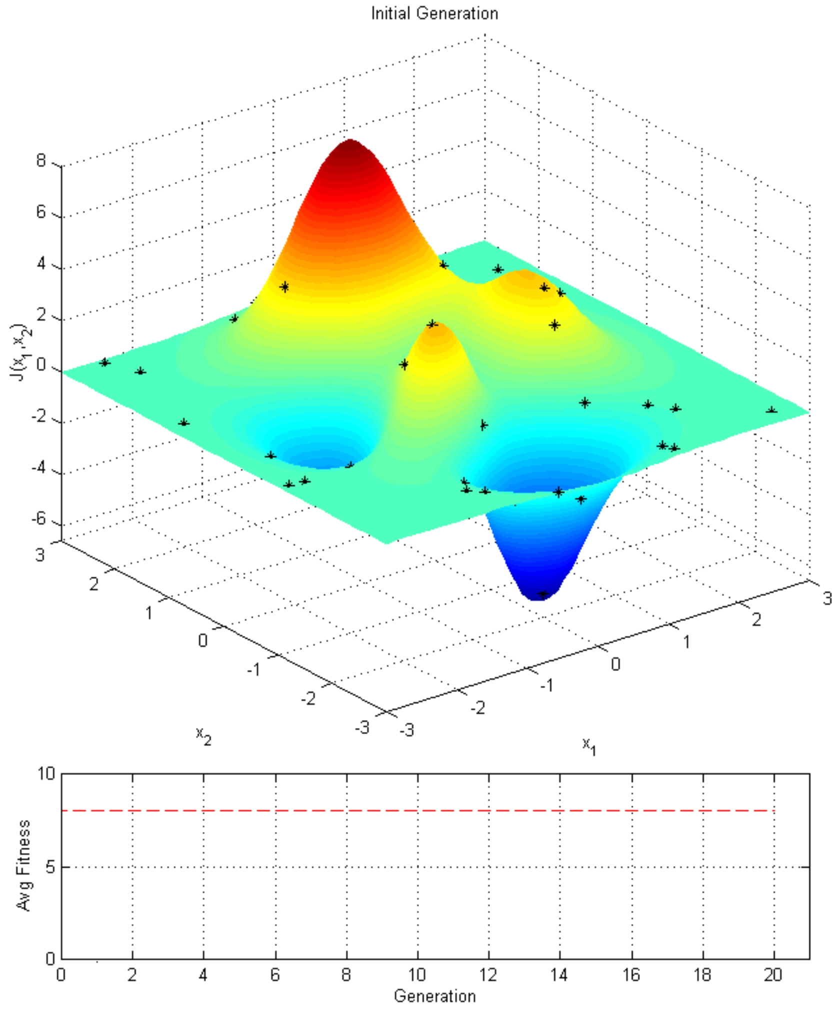
- A possible design of a system – as represented by its design vector **x -** can be considered as an individual who is fighting for survival within a larger population.

- Only the fittest survive – Fitness is assessed via objective function *J*.

MATLAB® GA demo ("peaks")

- Maximize "peaks" function
- Population size: 40
- Generations: 20
- Mutation Rate: 0.002

-Observe convergence
-Notice "mutants"
-Compare to gradient search



Natural Selection

Charles Darwin (1809-1882)

Controversial and very influential book (1859)
On the origin of species by means of natural selection, or the preservation of favored races in the struggle for life

Observations:
- Species are continually developing
- Homo sapiens sapiens and apes have common ancestors
- Variations between species are enormous
- Huge potential for production of offspring, but only a small/moderate percentage survives to adulthood

⇨ Evolution = natural selection of inheritable variations

Inheritance of Characteristics

Gregor Mendel (1822-1884)

Investigated the inheritance of characteristics ("traits")

Conducted extensive experiments with pea plants

Examined hybrids from different strains of plant

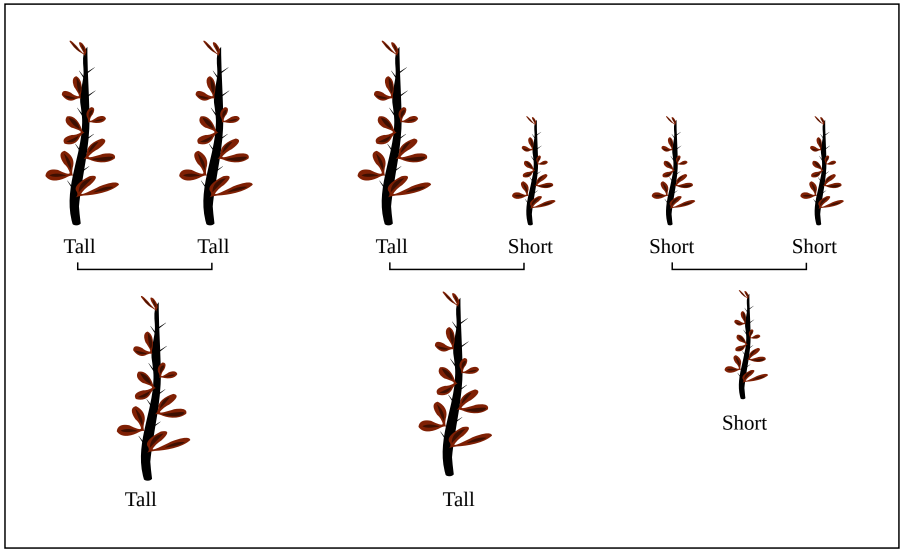
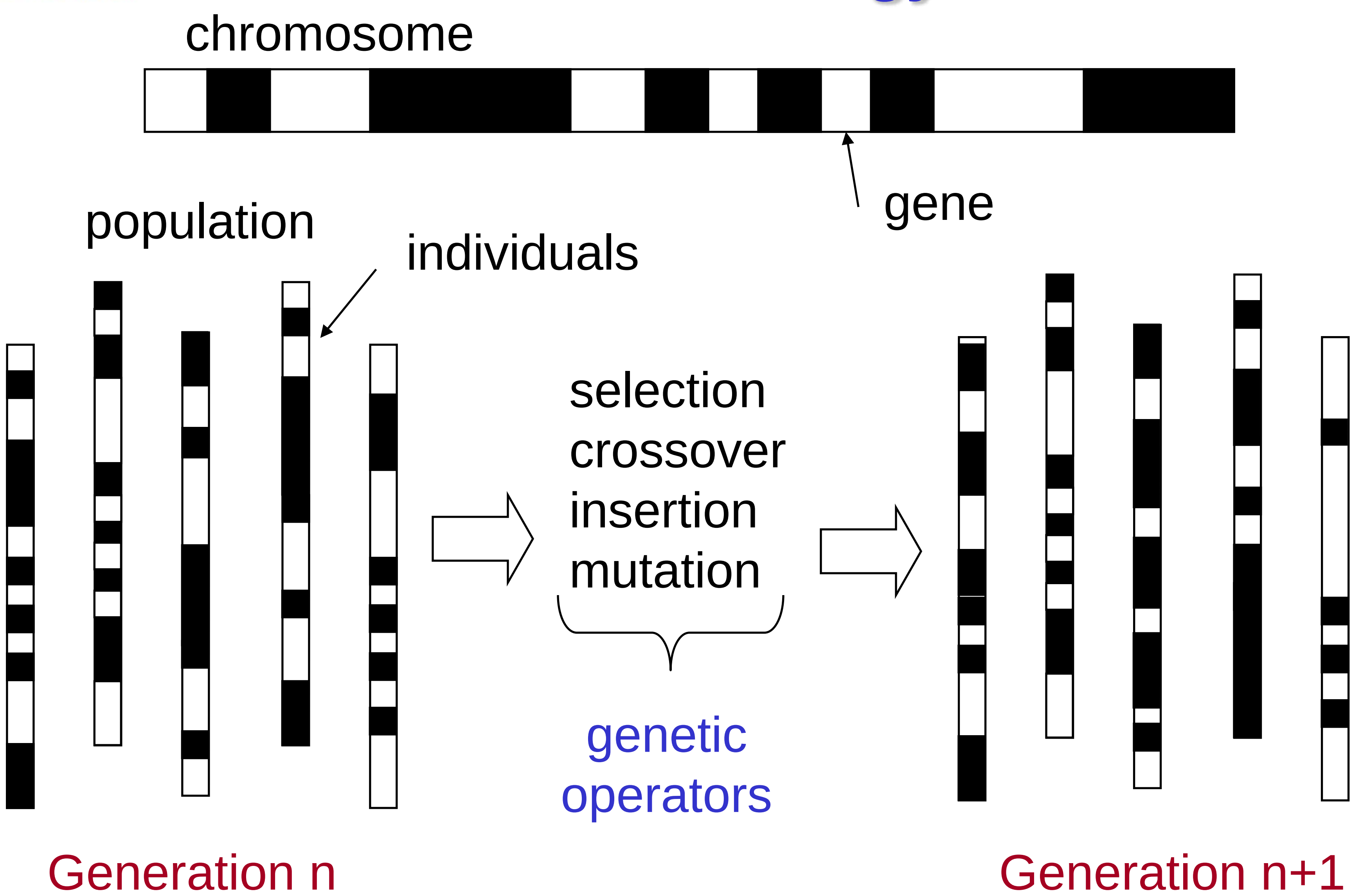


Image by MIT OpenCourseWare.

Character (gene) for tallness is dominant

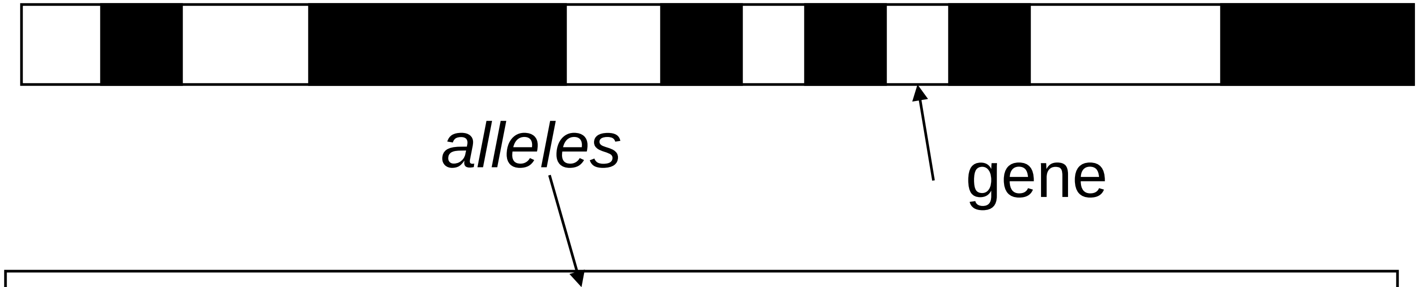
Character (gene) for shortness is recessive

GA Terminology



Chromosomes

Chromosome (string)



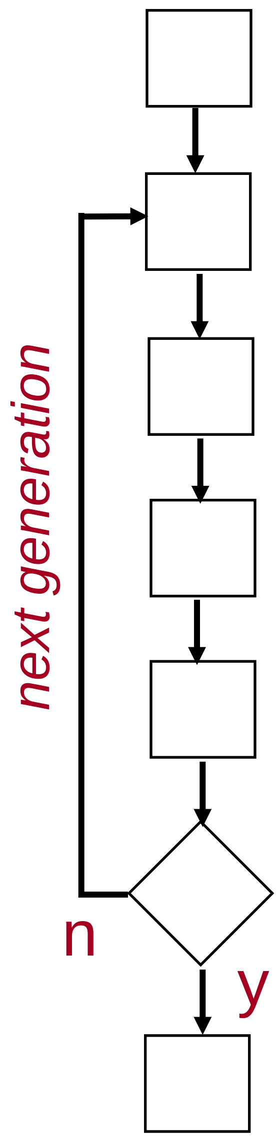
0 1 0 1 1 1 1 0 1 0 0 1 ….. 0 1

Each chromosome represents a solution, often using strings of 0’s and 1’s. Each bit typically corresponds to a gene. This is called binary encoding.

The values for a given gene are the alleles.

A chromosome in isolation is meaningless -
need decoding of the chromosome into phenotypic values

GA over several generations



Initialize Population (initialization)

Select individual for mating (selection)

Mate individuals and produce children (crossover)

Mutate children (mutation)

Insert children into population (insertion)

Are stopping criteria satisfied ?

Finish

Ref: Goldberg (1989)

"The GA Game"

Ca. 15 minutes

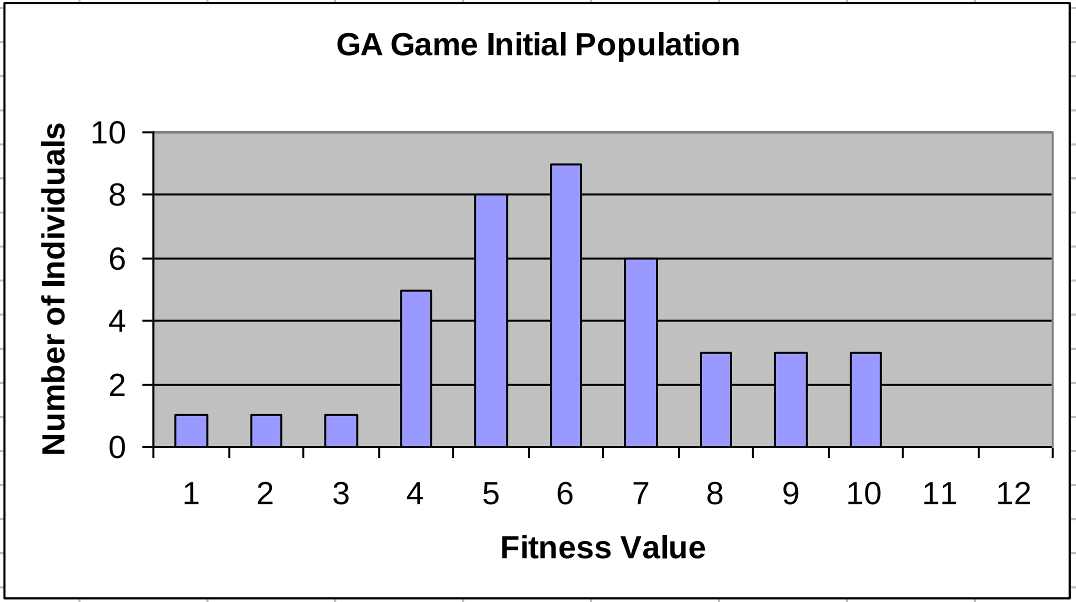
Population size: N=40

Mean Fitness: F=6.075

Generation 1:

(Fitness F = total number of 1's in chromosome)

1	1	1
2	1	2
3	1	3
4	5	20
5	8	40
6	9	54
7	6	42
8	3	24
9	3	27
10	3	30
11	0	0
12	0	0
	40	6.075



0 <= F <= 12

Goal: Maximize Number of "1"s

Creating a GA on Computer

(1) define the representation (encoding-decoding)
(2) define "fitness" function *F*
- incorporate feasibility (constraints) and objectives

(3) define the genetic operators
- initialization, selection, crossover, mutation, insertion

(4) execute initial algorithm run
- monitor average population fitness
- identify best individual

(5) tune algorithm
- adjust selection, insertion strategy, mutation rate

Encoding - Decoding

genotype

coded domain

phenotype

decision domain

Biology

UGCAACCGU

("DNA" blocks)

expression →

← *sequencing*

"blue eye"

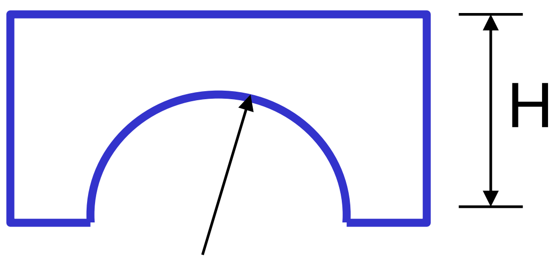
Design

10010011110

(chromosome)

decoding →

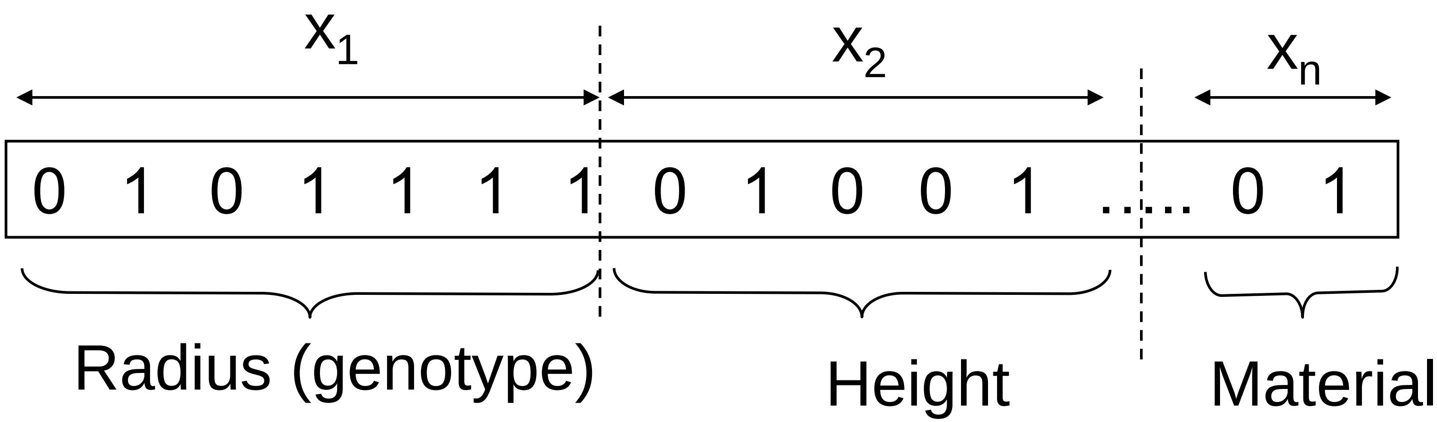
← *encoding*



Radius R=2.57 [m]

Genetic Code: (U,C,G,A are the four bases of the nucleotide building blocks of messenger-RNA): Uracil-Cytosin-Adenin-Guanin - A triplet leads to a particular aminoacid (for protein synthesis) e.g. UGG-tryptophane

Decoding



E.g. binary encoding of integers:

10100011

$(1{*}2^7+0{*}2^6+1{*}2^5+0{*}2^4+0{*}2^3+0{*}2^2+1{*}2^1+1{*}2^0)$

$128 + 0 + 32 + 0 + 0 + 0 + 2 + 1 = 163$

Coding and decoding MATLAB® functions available:
decode.m, encode.m

Binary Encoding Issues

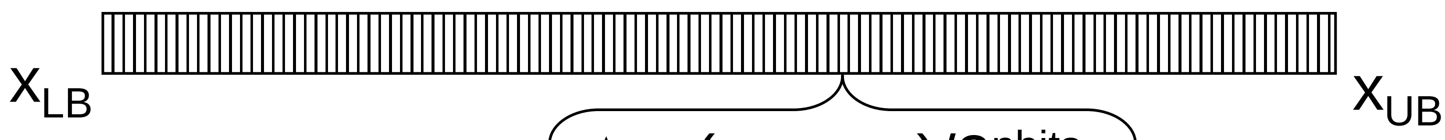
Number of bits dedicated to a particular design variable is very important.

Number of bits needed:

$$nbits = \left\lceil \frac{\ln\left(\frac{x_{UB} - x_{LB}}{\Delta x}\right)}{\ln 2} \right\rceil$$

Resolution depends on:
- upper and lower bounds x_{LB}, x_{UB}
- number of bits

$x \in$ □



$\Delta x = (x_{UB-} x_{LB})/2^{nbits}$

Example

```
[G]=encode(137.56,50,150,8)
G = 1     1     0     1     1     1     1     1
[X]=decode(G,50,150,8);
X = 137.4510
```

So $\Delta x = (150-50)/2^8 = 0.39$

Loss in precision !!!

Other Encoding Schemes

Not all GA chromosomes are binary strings
Can use a different ALPHABET for GA coding

Most common is binary alphabet {0,1}

can also have

The set of symbols is the "alphabet"

- ternary: {0,1,2} {A,B,C}
- quaternary: {0,1,2,3} {T,G,C,A} => biology
- integer: {1,2,….13,….}
-real valued: {3.456 7.889 9.112}
-Hexadecimal {1,2,..,A,B,C,D,E,F}

Used for Traveling Salesman Problem

A representation for the fire station location problem

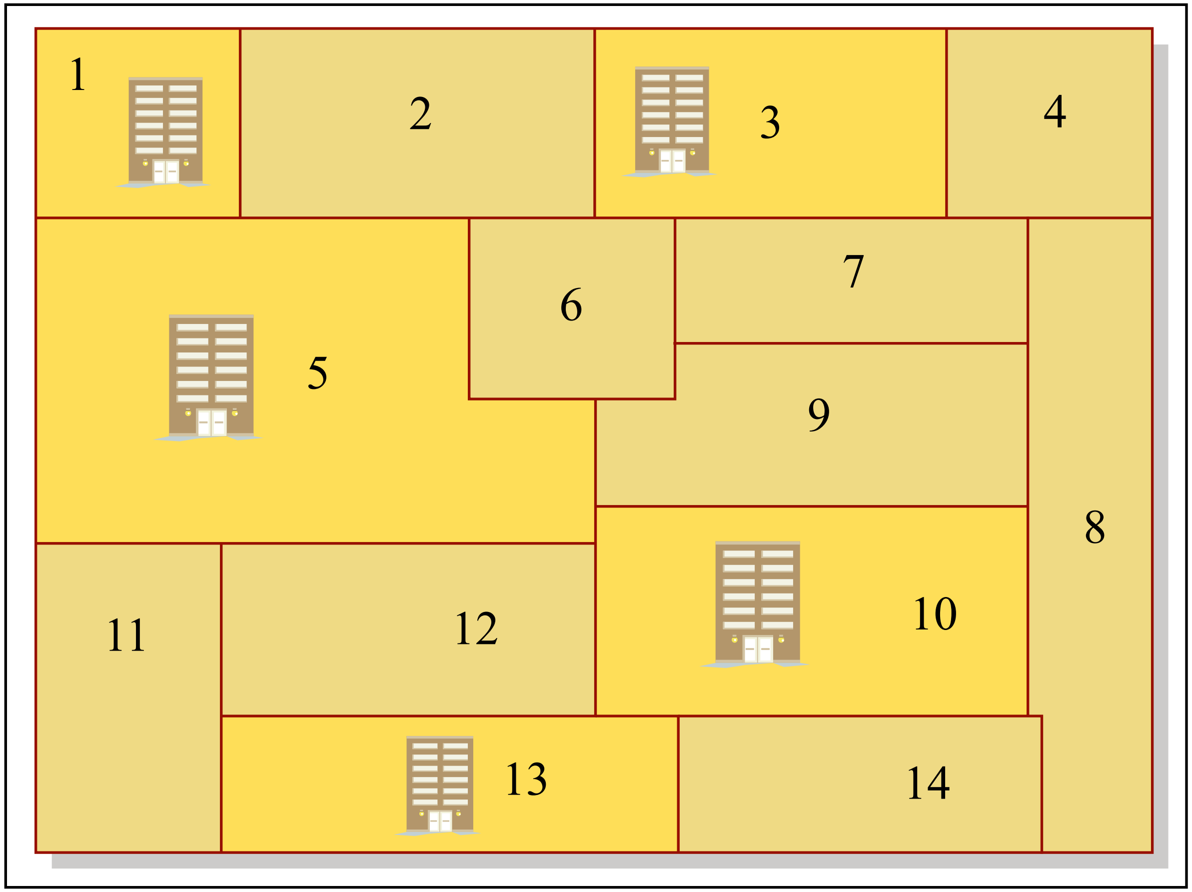


Image by MIT OpenCourseWare.

1 0 1 0 1 0 0 0 0 1 0 0 1 0

"1" represents a fire station

Fitness and Selection Probability

Typically, selection is the most important and most computationally expensive step of a GA.

$\boxed{01001110101}$ → decode → $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1.227 \\ \vdots \\ \text{Al-7075} \end{bmatrix}$

↓

Evaluate objective function — *simcode*

↓

$J = f\ \mathbf{x}$

← Map raw objective to Fitness ←

$F = f(J)$

↑

F drives probability of being selected

$P(selected) \propto F$ → yes / no

Fitness Function

- Choosing the right fitness function is very important, but also quite difficult
- GAs do not have explicit “constraints”
- Constraints can be handled in different ways:
  - via the fitness function – penalty for violation
  - via the selection operator (“reject constraint violators”)
  - implicitly via representation/coding e.g. only allow representations of the TSP that correspond to a valid tour
  - Implement a repair capability for infeasible individuals

⇨ Choosing the right fitness function: an important genetic algorithm design Issue

Maximization vs. Minimization

There are many ways to convert a minimization problem to a maximization problem and vice-versa:

- N-obj
- 1/obj
- -obj

Selection by Ranking

- Goal is to select parents for crossover
- Should create a bias towards more fitness
- Must preserve diversity in the population

(1) Selection according to RANKING

Example: Let $D = \sum_{j \in P} 1/j$

select the kth most fit member of a population

to be a parent with probability $P_k = \left(\frac{1}{k}\right) D^{-1}$

Better ranking has a higher probability of being chosen, e.g. 1st $\propto$ 1, 2nd $\propto$ 1/2, 3rd $\propto$ 1/3 ...

Selection by Fitness

(2) Proportional to FITNESS Value Scheme

Example: Let $\bar{F} = \sum_{j \in P} Fitness\ \ j$

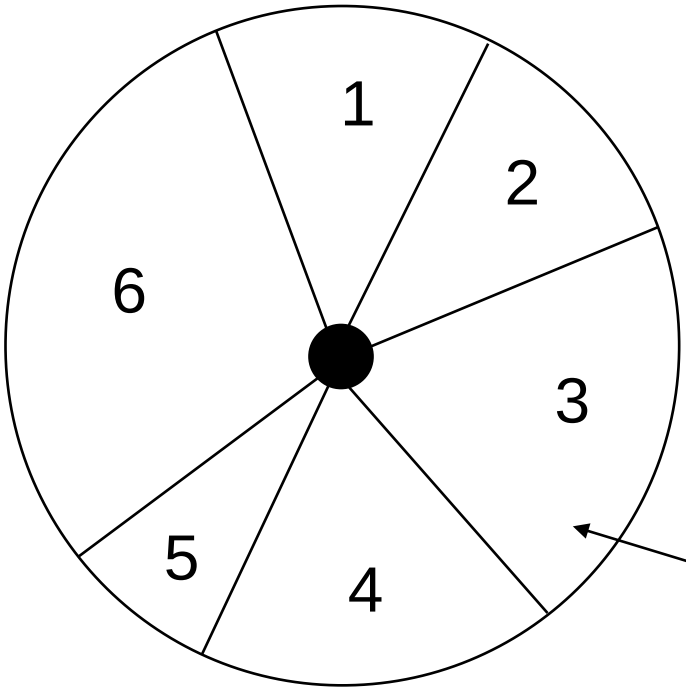
select the k[th] most fit member of a population

to be a parent with probability $P_k = Fitness(k) \cdot \bar{F}^{-1}$

⇒ Probability of being selected for crossover is <u>directly</u> proportional to raw fitness score.

Roulette Wheel Selection

Roulette Wheel Selection



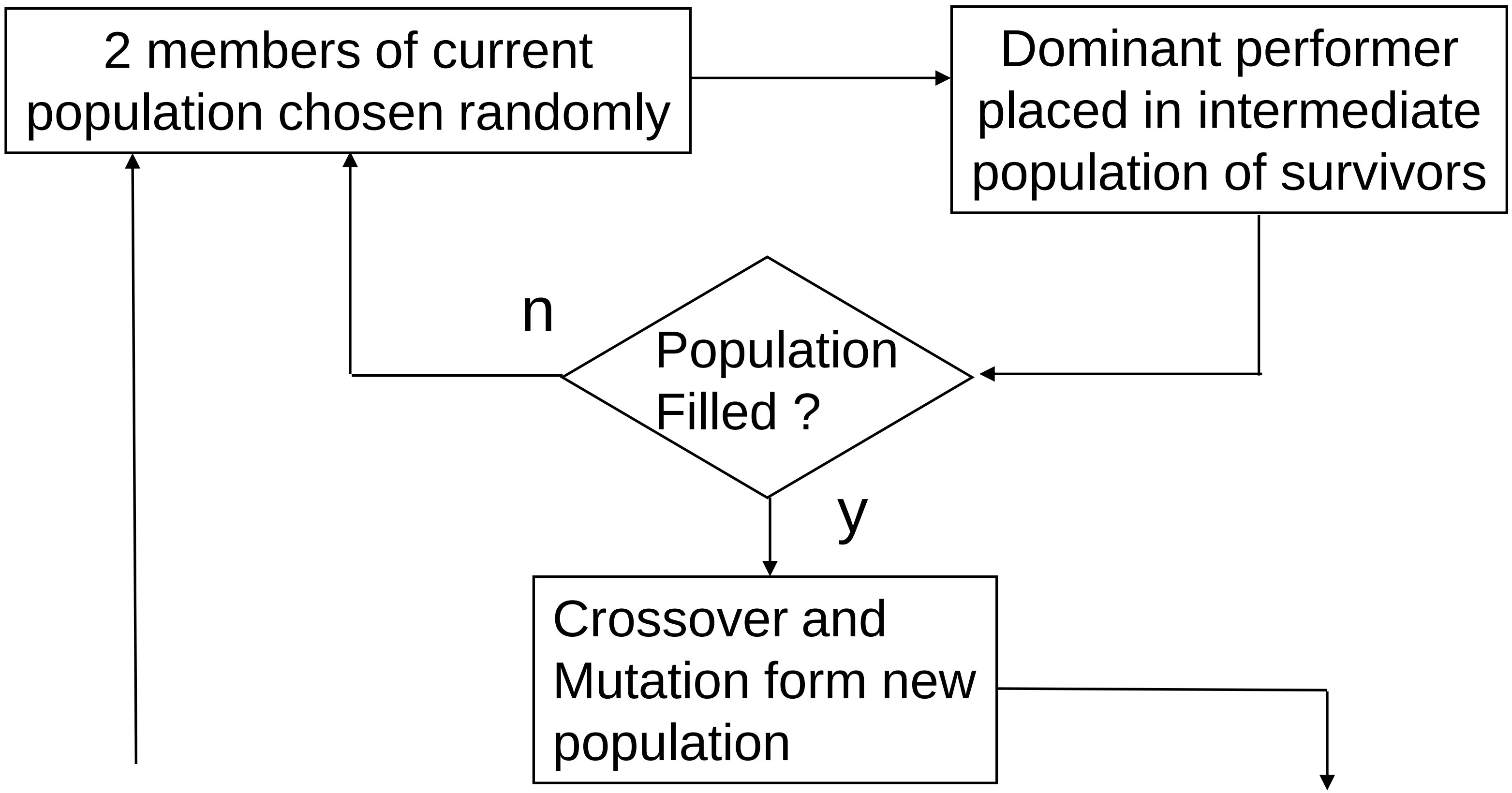
Probabilistically select individuals based on some measure of their performance.

Sum Sum of individual's selection probabilities

3rd individual in current population mapped to interval [0,*Sum*]

Selection: generate random number in [0,*Sum*]
Repeat process until desired # of individuals selected
Basically: stochastic sampling with replacement (SSR)

Tournament Selection



Old Population	Fitness
101010110111	8
100100001100	4
001000111110	6

Survivors	Fitness
101010110111	8
001000111110	6
101010110111	8

Crossover

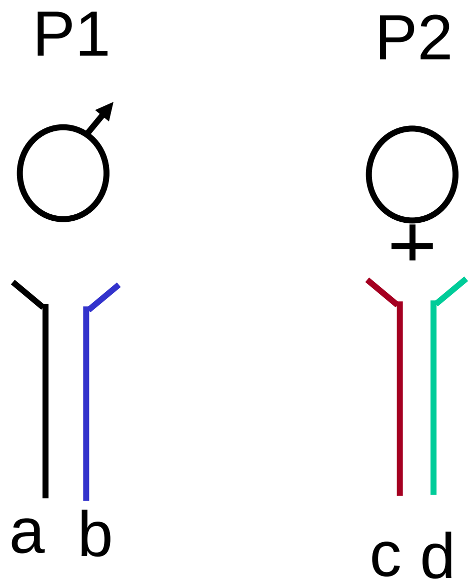
♂	0 1 0 1 1 1 1 0 1 1 1 1 ….. 1 1	P1
♀	1 1 1 0 0 1 0 0 0 1 0 1 ….. 0 0	P2

⇩ crossover

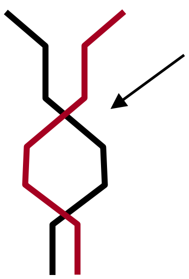
O1	?
O2	?

Question: How can we operate on parents P1 and P2 to create offspring O1 and O2 (same length, only 1's and 0's)?

Crossover in Biology



Child



This is where the word crossover comes from

ac

Crossover produces either of these results for each chromosome

ac OR ad OR bc OR bd

Crossover Operator (I)

Crossover (mating) is taking 2 solutions, and creating 1 or 2 more

Classical: single point crossover

crossover point

	Parents		Children	
P1	0 1 1 | 0 1		0 1 1 | 1 1	O1
P2	1 0 0 | 1 1		1 0 0 | 0 1	O2

The parents

The children ("offspring")

Crossover Operator (II)

More on 1-point crossover ….

P1	0	1	1 |	0	1		0	1	1 |	1	1	C1
P2	1	0	0 |	1	1		1	0	0 |	0	1	C2

l=length of chromosome

i=3

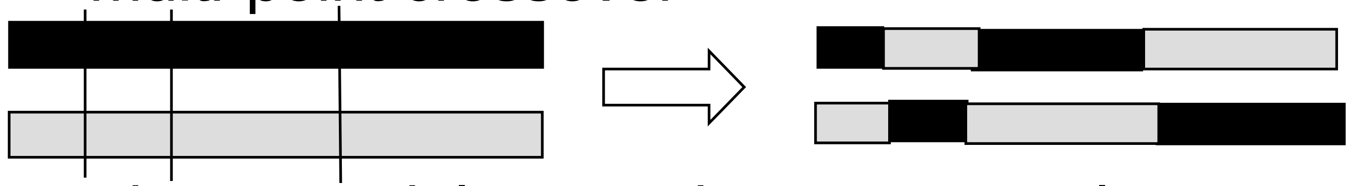
i=3 *l*=5

A crossover bit "i" is chosen (deliberately or randomly), splitting the chromosomes in half.

Child C1 is the 1st half of P1 and the 2nd half of P2
Child C2 is the 1st half of P2 and the 2nd half of P1

Crossover Operator (III)

- One can also do a 2-point crossover or a multi-point crossover



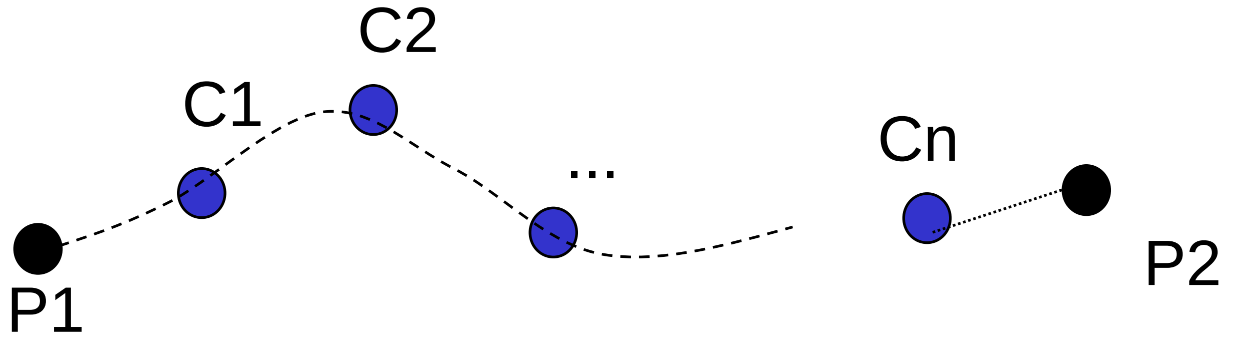
- The essential aspect is to create at least one child (solution/design) from two (or more) parent (solutions/designs)
  - there are many solutions to do this

Some crossover operations:

- single point, versus multiple point crossover
- path re-linking

Path Relinking

- Given Parents P1 and P2
- Create a sequence of children
  - The first child is a neighbor of P1
  - Each child is a neighbor of the previous child
  - The last child is a neighbor of P2



Example: Path Relinking

Parents

P1 | 1 0 0 1 0 0 1 | and | 0 0 1 0 1 0 0 | P2

Children

| 1 0 0 1 0 0 0 |
| 1 0 0 1 1 0 0 |
| 1 0 0 0 1 0 0 |
| 1 0 1 0 1 0 0 |

Create a path of children, then select the best one.

Good approach, but solutions tend to be interpolations of initial population.

Some Insertion Strategies

- Can replace an entire population at a time (go from generation k to k+1 with no survivors)
  - select N/2 pairs of parents
  - create N children, replace all parents
  - polygamy is generally allowed

N = # of members in population if steady state

- Can select two parents at a time
  - create one child
  - eliminate one member of population (weakest?)

- "Elitist" strategy
  - small number of fittest individuals survive unchanged

- "Hall-of-fame"
  - remember best past individuals, but don't use them for progeny

Initialization

Somehow we need to create an initial population of solutions to start the GA. How can this be done?

- Random initial population, one of many options
- Use random number generator to create initial population (caution with seeds !)
- Typically use uniform probability density functions (pdf's)
- Typical goal: Select an initial population that has both quality and diversity

Example:

N_{ind} - size of binary population
L_{ind} - Individual chromosome length

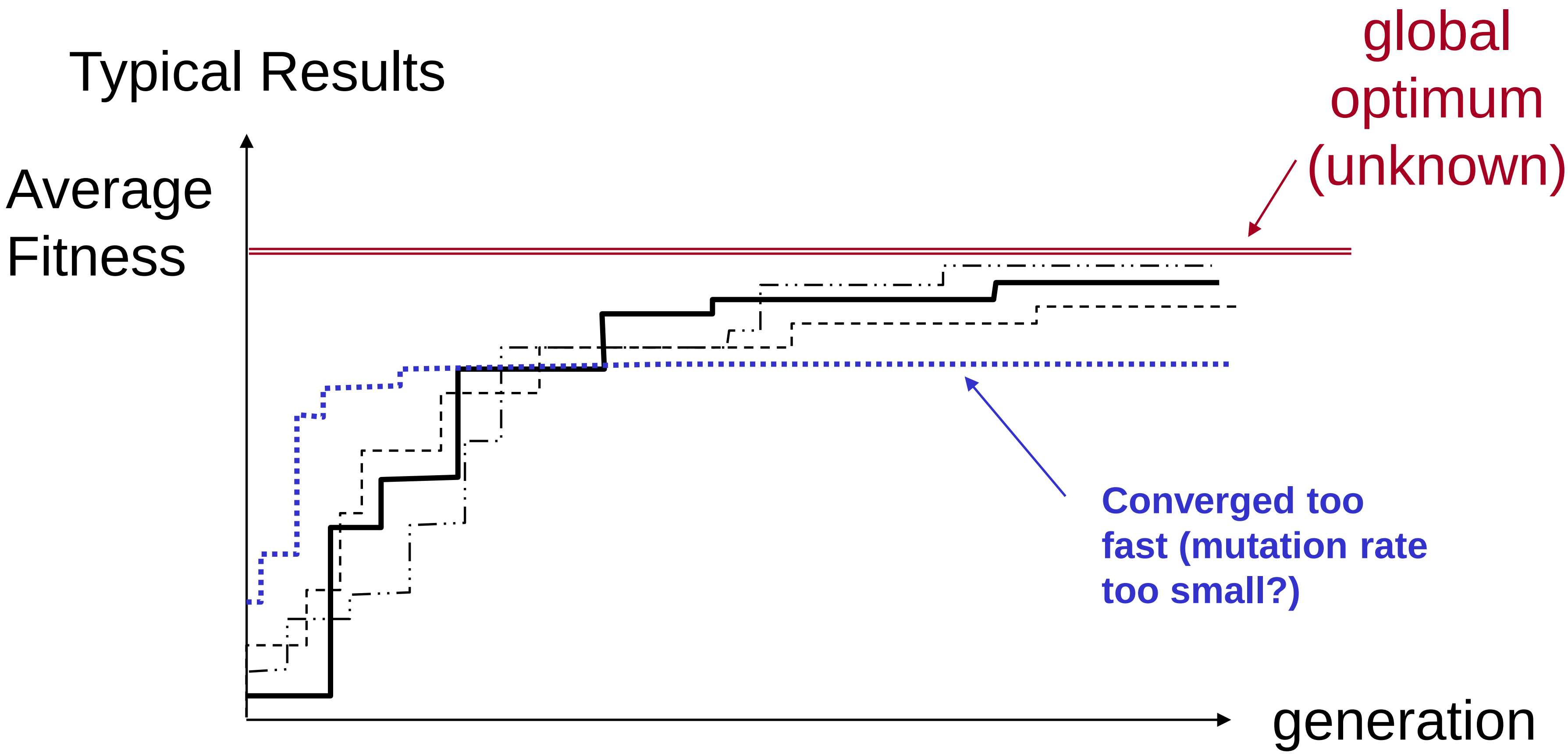
```
round(rand(1,6))    >>  1 1 1 1 0 0
```

Need to generate N_{ind} x L_{ind} random numbers from {0,1}

Rule of thumb: Population Size at Least $N_{ind} \sim 4\ L_{ind}$

GA Convergence

Typical Results



<u>Average</u> performance of individuals in a population is expected to increase, as good individuals are preserved and bred and less fit individuals die out.

GA Stopping Criteria

Some options:

- X number of generations completed - typically O(100)
- Mean deviation in performance of individuals in the population falls below a threshold σ_J<x (genetic diversity has become small)
- Stagnation - no or marginal improvement from one generation to the next: $(J_{n+1}-J_n)<X$

GAs versus other methods

16.888
ESD.77

Differ from traditional search/optimization methods:

- GAs search a population of points in parallel, not only a single point
- GAs use probabilistic transition rules, not deterministic ones
- GAs work on an encoding of the design variable set rather than on the variables themselves
- GAs do not require derivative information or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search

Next Lecture

- Speciality GA's
- Particle Swarm Optimization (PSO)
- Tabu Search (TS)
- Selection of Optimization Algorithms
  - Which algorithm is most suited to my problem?
- Design Optimization Applications

Book References

Holland J., “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975

Goldberg, D.E.,” Genetic Algorithms in Search, Optimization and Machine Learning”, Addison Wesley, 1989

ESD.77 / 16.888 Multidisciplinary System Design Optimization
Spring 2010