

Multidisciplinary System Design Optimization

Heuristic Techniques: A Basic Introduction to Genetic Algorithms

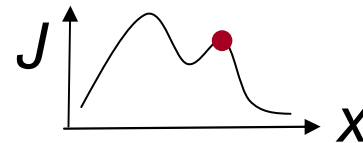
Lecture 11

March 8, 2004

Olivier de Weck

Main Motivation for Heuristic Techniques:

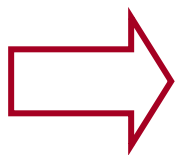
(1) To deal with local optima and not get trapped in them



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

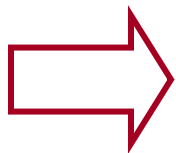
$$x_i \notin \mathbb{R}$$

$$x_i = \{1, 2, 3, 4, 5\}, x_i = \{'A', 'B', 'C'\} x_i = \{\text{true}, \text{false}\}$$



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

- Genetic Algorithms (Holland – 1975)
 - Inspired by genetics and natural selection
- Simulated Annealing (Kirkpatrick – 1983)
 - Inspired by molecular dynamics – energy minimization
- Particle Swarm Optimization (Eberhart and Kennedy - 1995)
 - Inspired by the social behavior of swarms of insects or flocks of birds



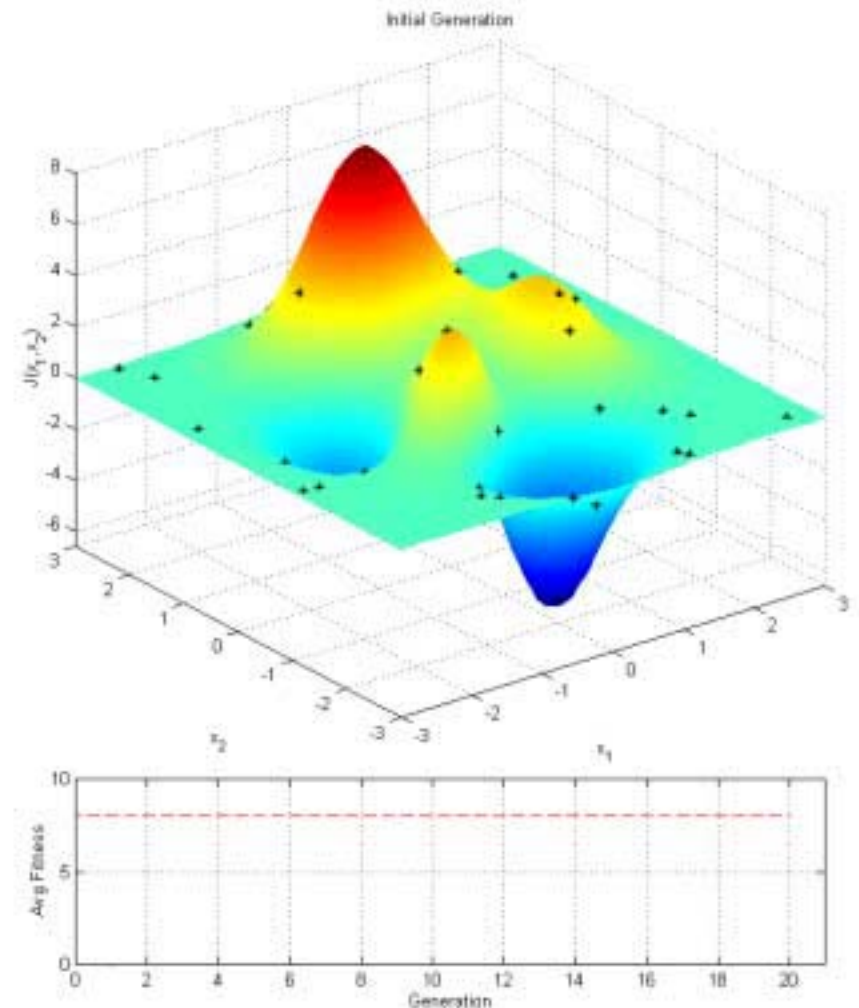
These techniques all use a combination of randomness and heuristic “rules” to guide the search for global maxima or minima

Part 1 - Introduction

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

- 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 \mathbf{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 .

- Maximize “peaks” function
 - Population size: 40
 - Generations: 20
 - Mutation Rate: 0.002
-
- Observe convergence
 - Notice “mutants”
 - Compare to gradient search



Charles Darwin (1809-1882)

Extremely controversial and influential book (1859)
On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life

Observations:

- Species are continually developing
- Homo sapiens sapiens comes from ape-like stock
- Variations between species are enormous
- Huge potential for production of offspring, but only a small percentage survives to adulthood



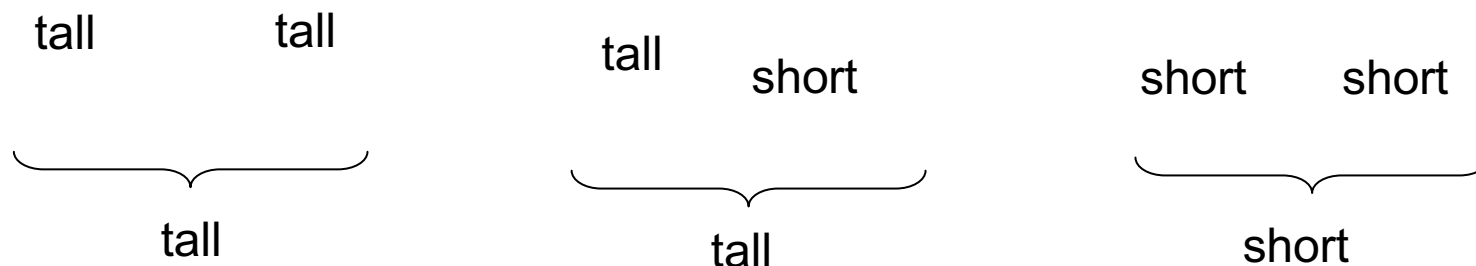
Evolution = natural selection of inheritable variations

Gregor Mendel (1822-1884)

Investigated the inheritance of characteristics (“traits”)

Conducted extensive experiments with pea plants

Examined hybrids from different strains of plant



Character (gene) for tallness is dominant
Character (gene) for shortness is recessive

... the great unknown Walter Sutton (1877-1916)

- Discovered that **genes are part of the chromosomes** inside the nucleus of cells

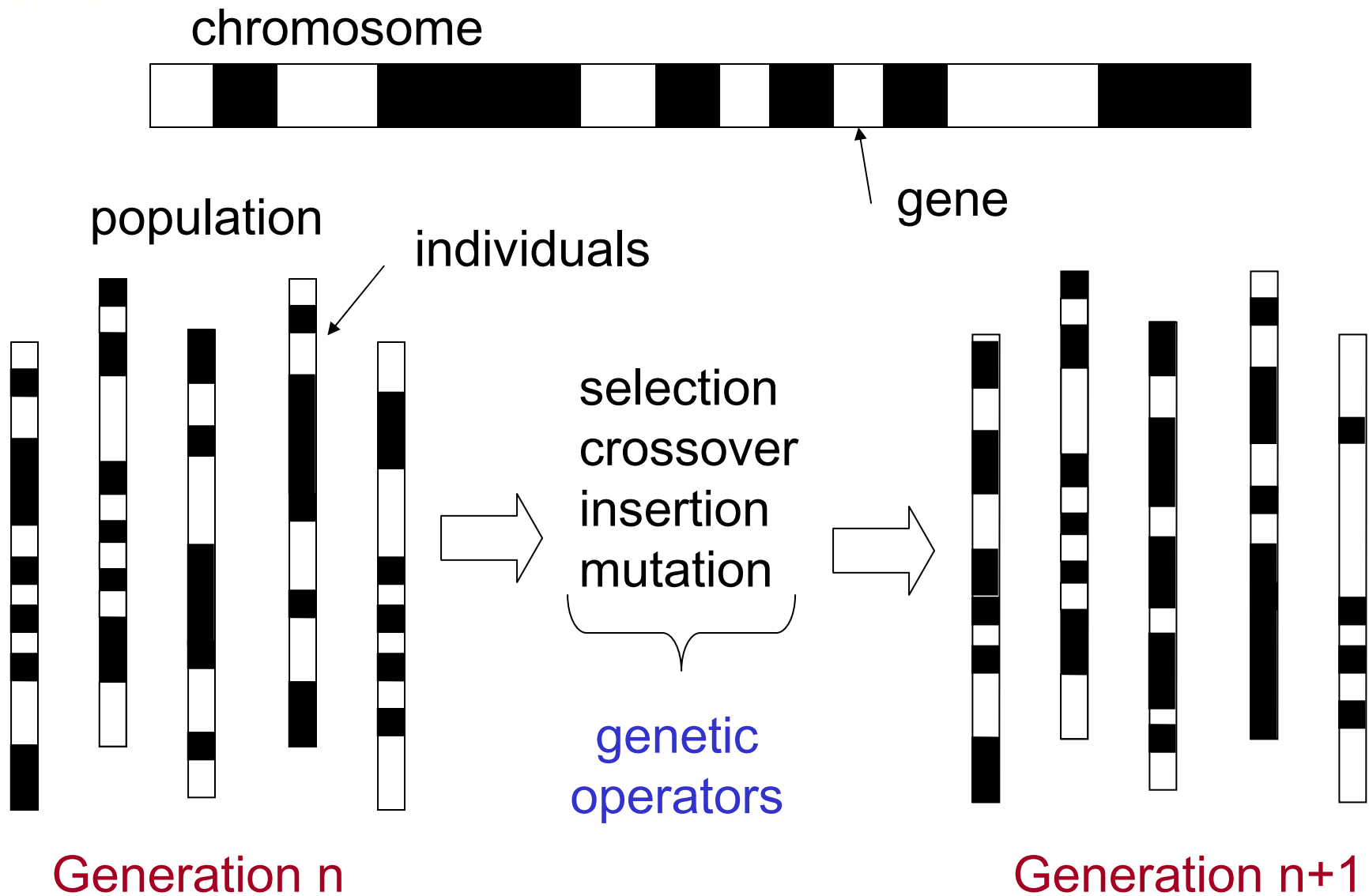
... still unexplained ... Darwin predicted continuous variation within species. In nature we often observe discontinuous variations

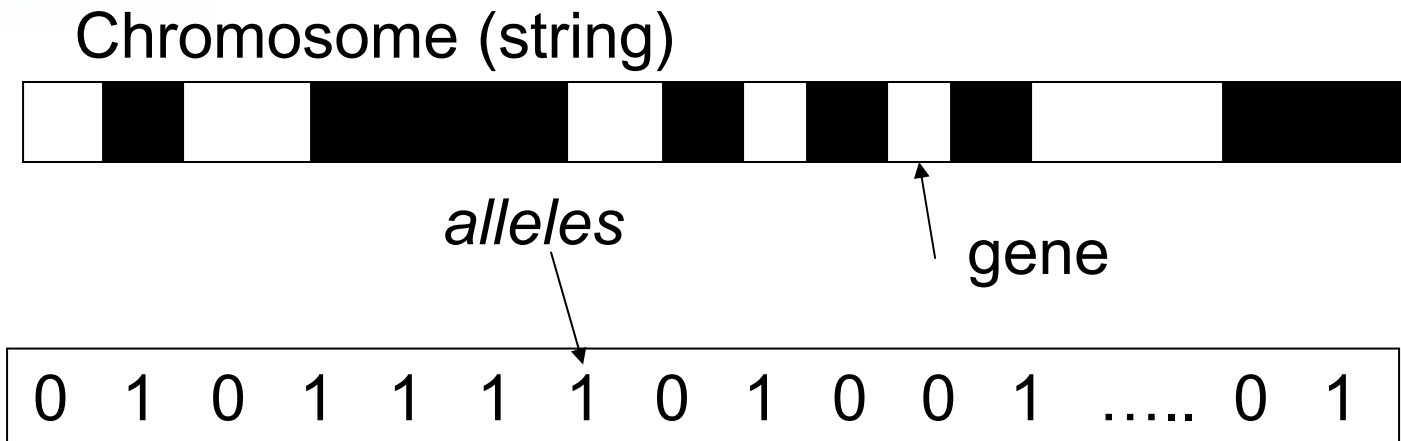
... strikingly different, unexpected variants sometimes appear, how to explain ???

Hugo de Varis (1848-1935) developed a **theory of mutation** - at first seemingly at odds with Darwinism



How has this dilemma been resolved?

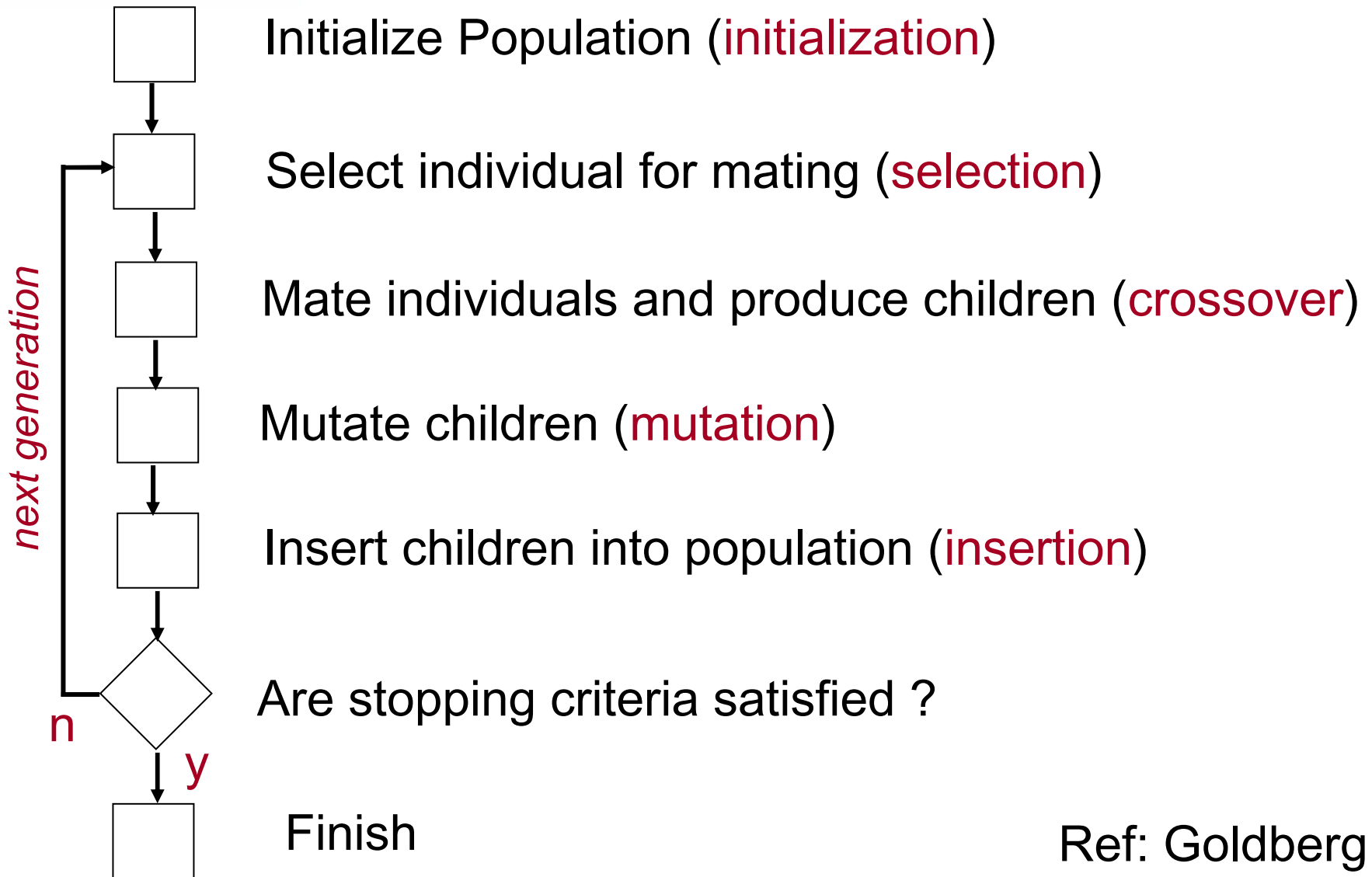




Each chromosome represents a solution, often using strings of 0's and 1's. Each bit typically corresponds to a gene. This is called binary encoding.

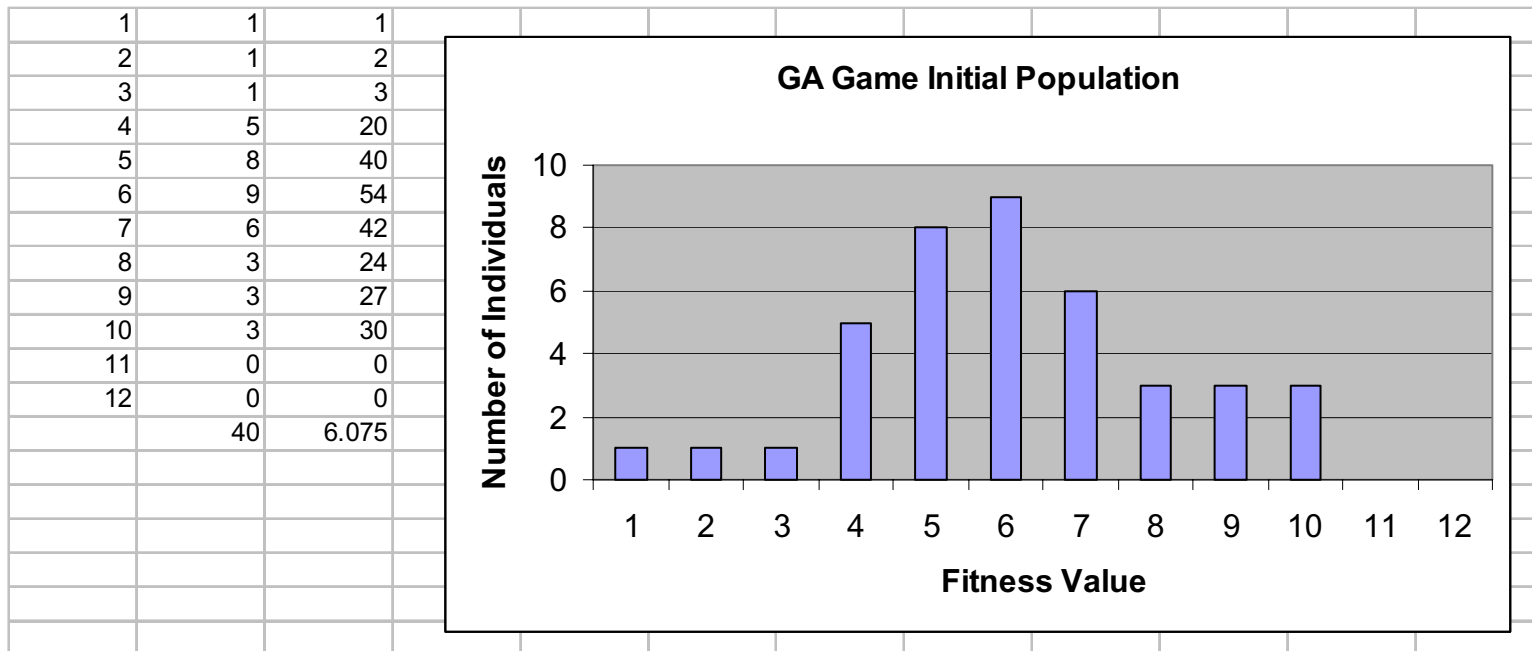
The values for a given gene are the alleles.

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

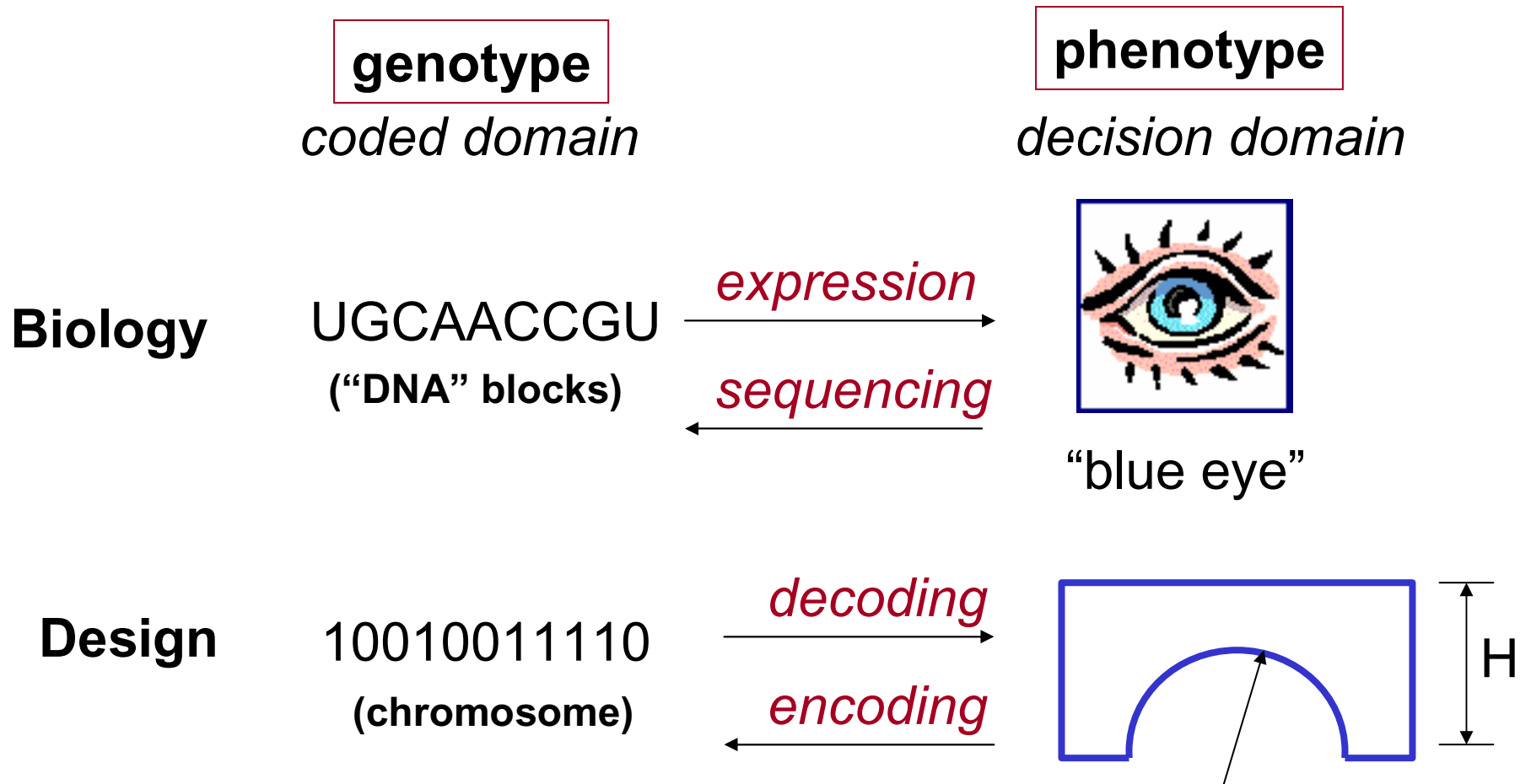


Ref: Goldberg

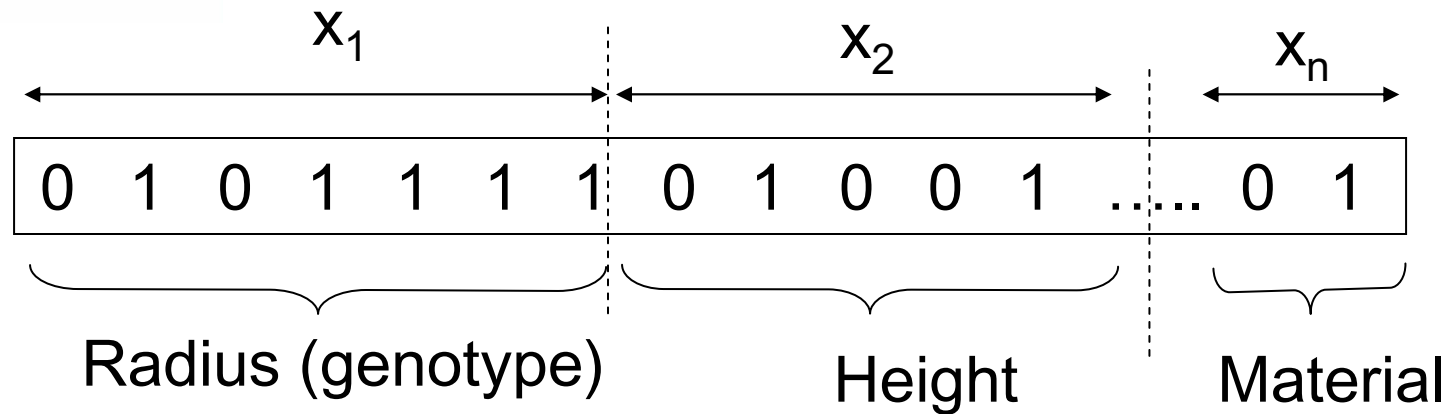
Ca. 15 minutes

Population size: $N=40$ Mean Fitness: $F=6.075$ **Generation 1:**(Fitness F = total number of 1's in chromosome) $0 \leq F \leq 12$ **Goal: Maximize Number of “1”s**

- (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



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



E.g. binary encoding of integers:

10100011

$$(1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0)$$

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

Coding and decoding MATLAB functions available:

decode.m, encode.m

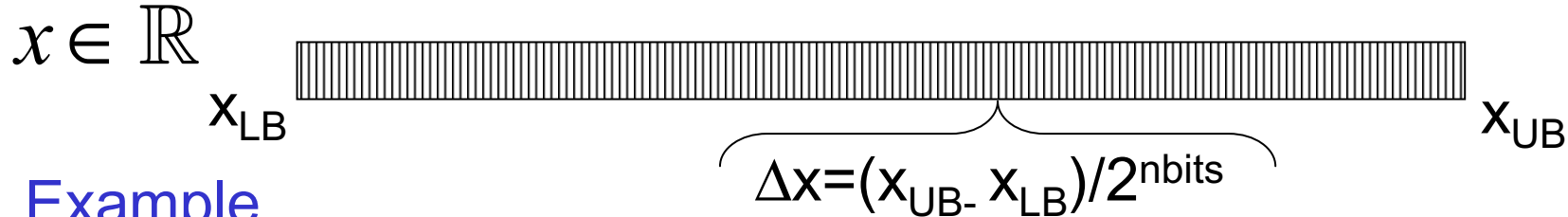
Number of bits dedicated to a particular design variable is very important.

Number of bits needed:

Resolution depends on:

- upper and lower bounds x_{LB} , x_{UB}
- number of bits

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



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 !!!

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

Most common is binary alphabet $\{0,1\}$

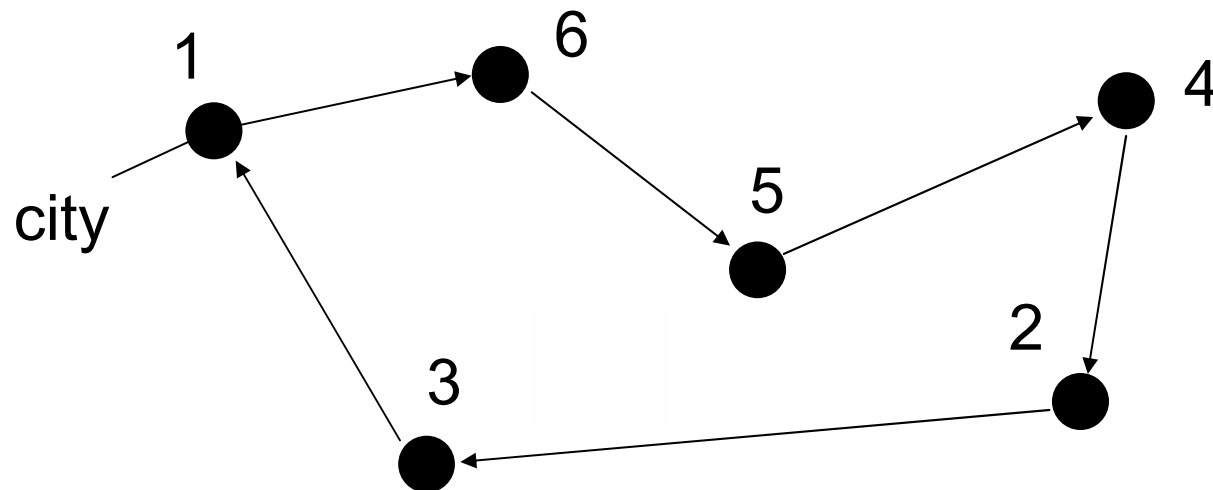
can also have

The set of symbols
is the “alphabet”

- ternary: $\{0,1,2\}$ $\{A,B,C\}$
- quaternary: $\{0,1,2,3\}$ $\{T,G,C,A\} \Rightarrow$ biology
- integer: $\{1,2,\dots,13,\dots\}$
- real valued: $\{3.456\ 7.889\ 9.112\}$
- Hexadecimal $\{1,2,\dots,A,B,C,D,E,F\}$

Used for Traveling
Salesman Problem

Two representations of the TSP



The arcs

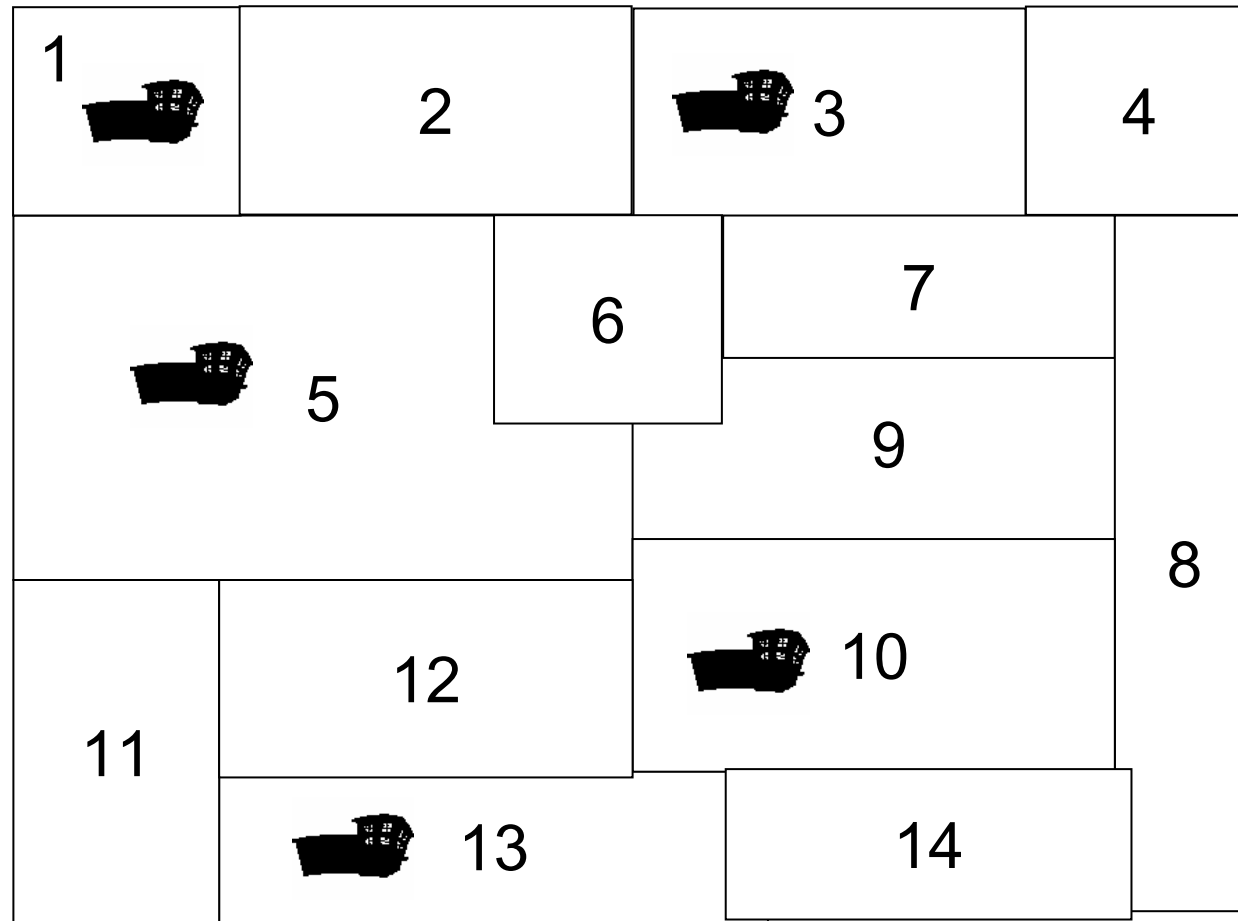
1	2	3	4	5	6
6	3	1	2	4	5

The ordering

1 6 5 4 2 3

Same problem,
but two different
chromosome
representations

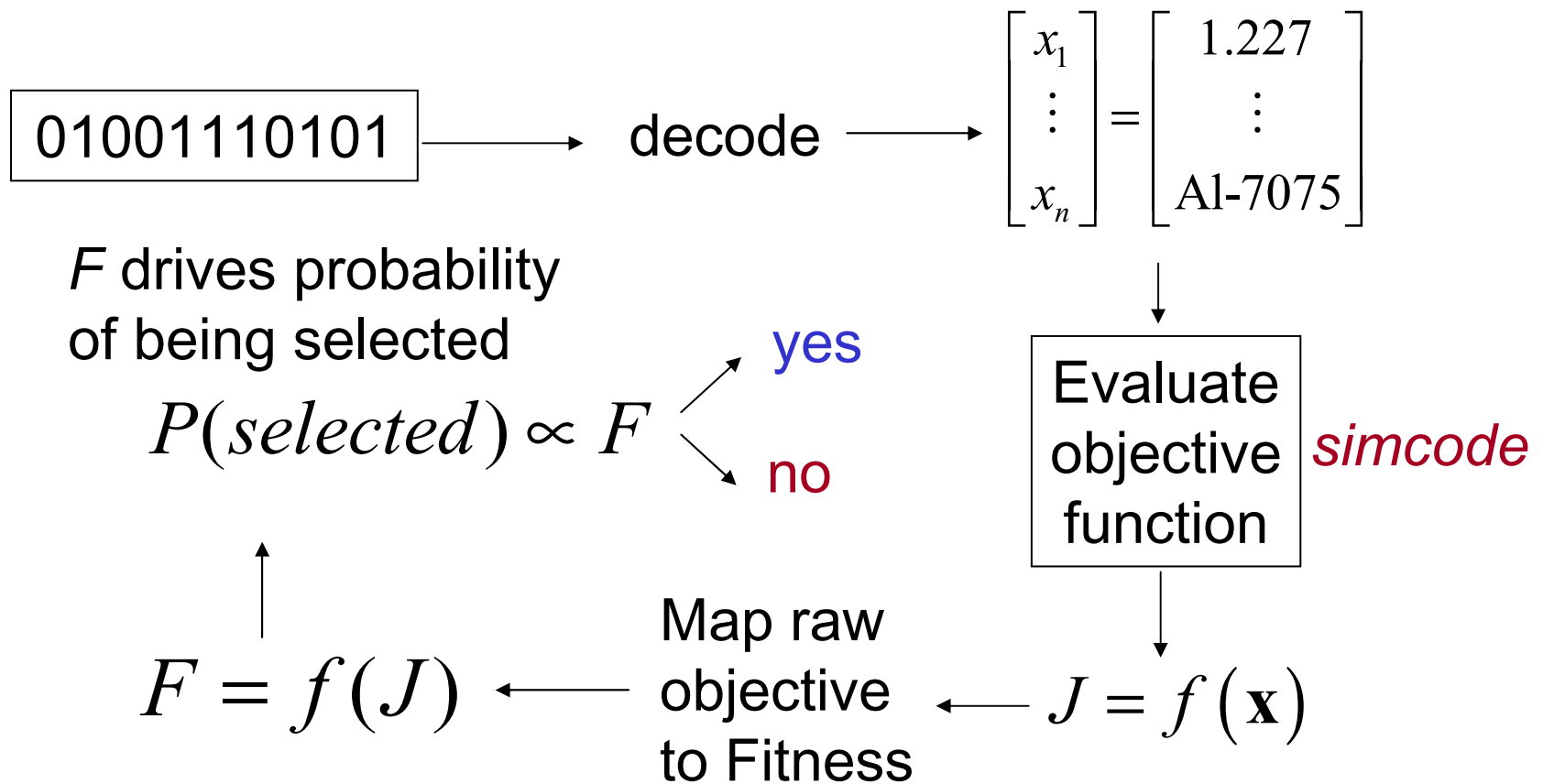
A representation for the fire station location problem

16.888
ESD.77

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

“1” represents a fire station

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



- Each chromosome has a “fitness”
- The objective function (value) is usually mapped into the fitness of each individual
- For the TSP the fitness is usually the cost of the tour (time, distance, price)
- The fitness for the fire station problem should incorporate feasibility
 - Example: $\text{fitness} = K - \# \text{ of fire stations} - \# \text{ of uncovered districts}$

- 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:
 - implicitly 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



Choosing the right fitness function: an important Genetic Algorithm Design Issue

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

- $N\text{-obj}$
- $1/\text{obj}$
- $-\text{obj}$

- 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 k^{th} most fit member of a population

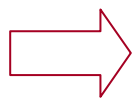
to be a parent with probability $P_k = \left(\frac{1}{k}\right) D^{-1}$



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

(2) Proportional to FITNESS Value Scheme

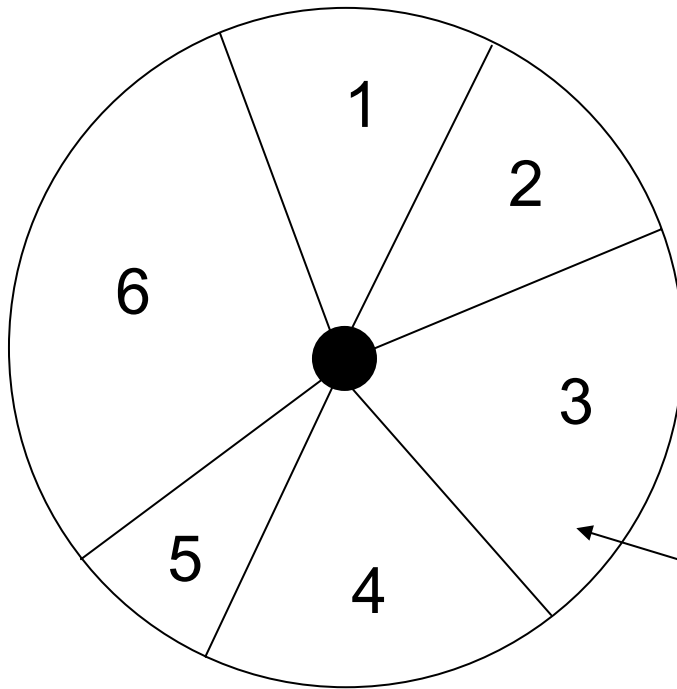
Example: Let $\bar{F} = \sum_{j \in P} \text{Fitness}(j)$
select the k^{th} most fit member of a population
to be a parent with probability $P_k = \text{Fitness}(k) \cdot \bar{F}^{-1}$



Probability of being selected for crossover is directly proportional to raw fitness score.

This scheme tends to favor the fittest individuals in a population more than the ranking-scheme, faster convergence, but can also be a disadvantage.

Roulette Wheel Selection

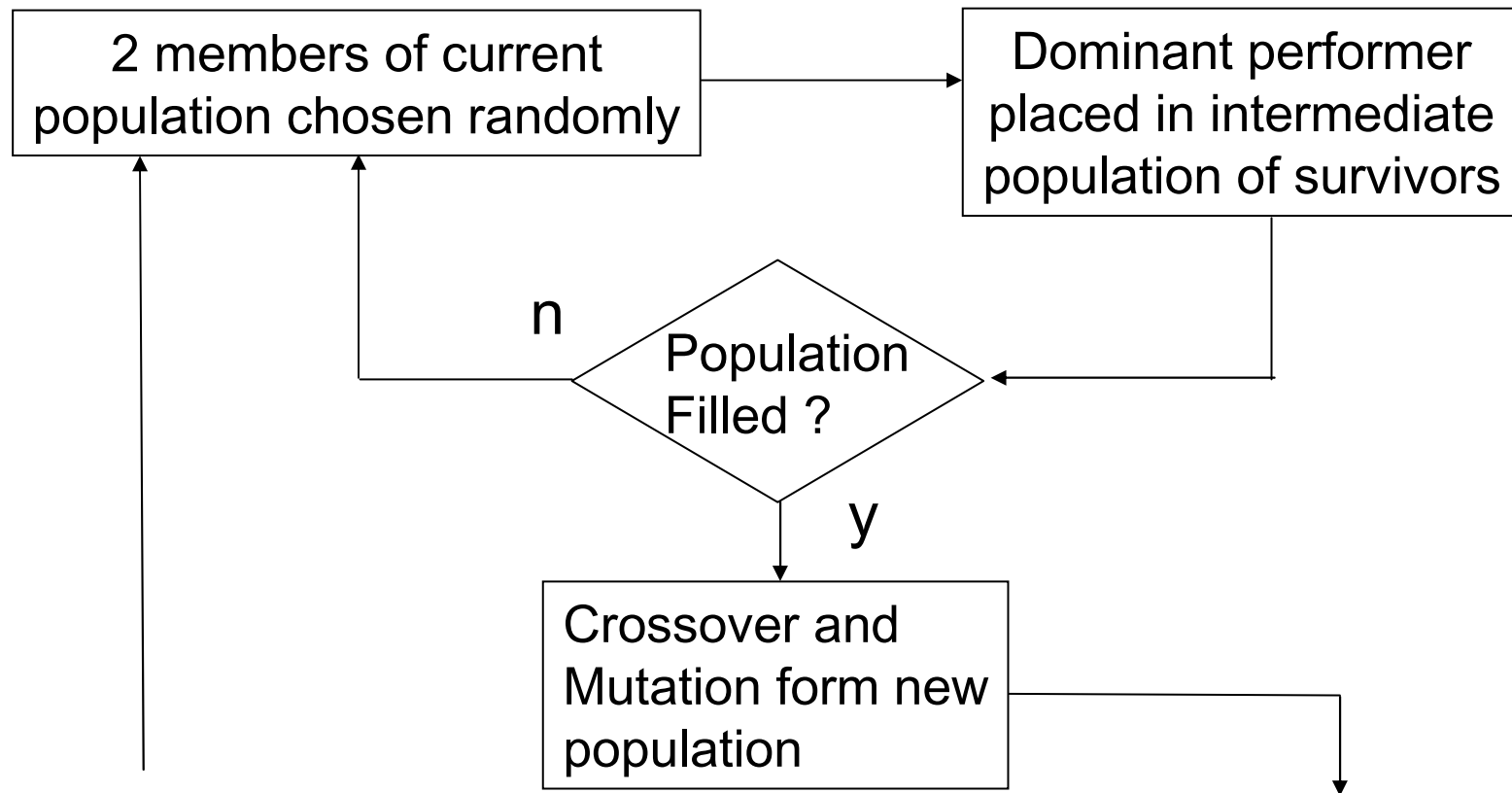


Probabilistically select individuals based on some measure of their performance.

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)

**Old Population** **Fitness**

101010110111	8	}
100100001100	4	
001000111110	6	

Survivors **Fitness**

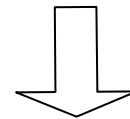
101010110111	8
001000111110	6
101010110111	8



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

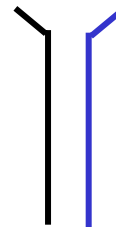
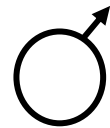
?

O2

?

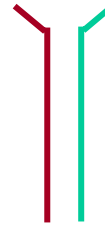
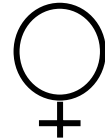
Question: How can we operate on parents P1 and P2 to create offspring O1 and O2 (same length, only 1's and 0's)?

P1



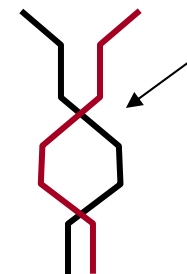
a b

P2



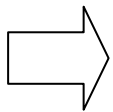
c d

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

Classical: **single point crossover**

P1	0	1	1		0	1
P2	1	0	0		1	1

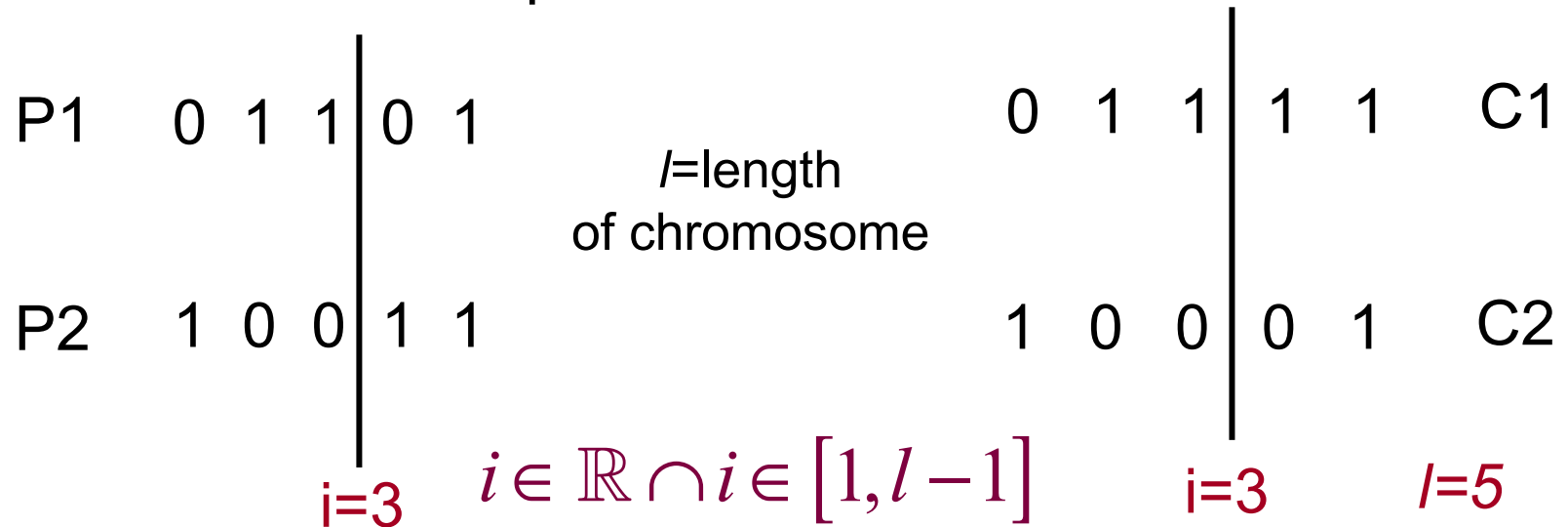
The parents

crossover point

0	1	1		1	1	O1
1	0	0		0	1	O2

The children
("offspring")

More on 1-point crossover

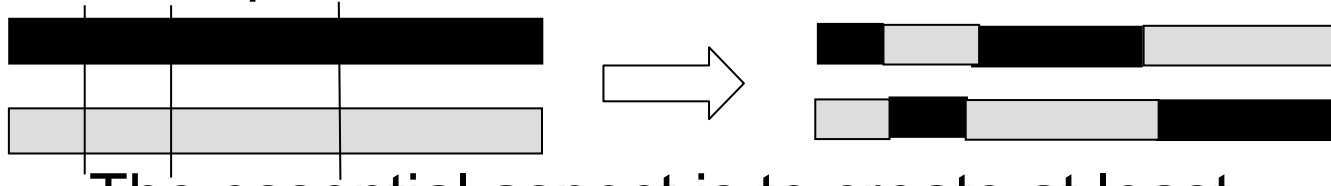


A crossover bit “ i ” is chosen (deliberately or randomly), splitting the chromosomes in half.

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

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

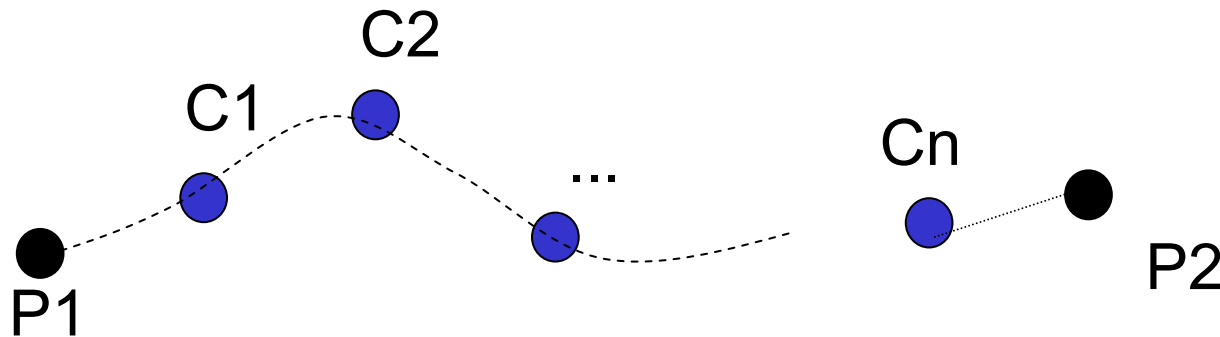


- The essential aspect is to create at least one child (solution/design) from two (or more) parent (solutions/designs)
 - there are many solutions to do this
 - do not necessarily have to do crossover, and do crossover with a probability P_x after pairs are chosen

Some crossover operations:

- single point, versus multiple point crossover
- path relinking
- permutation operators (list operators), incl. Random keys approach

- 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



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.

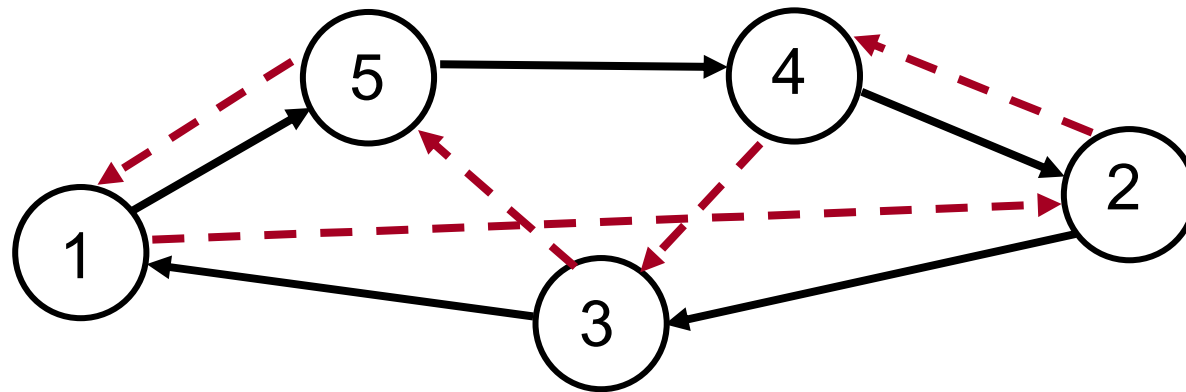
Route A

1	2	3	4	5
5	3	1	2	4



Route B

1	2	3	4	5
2	4	5	3	1



Problem:

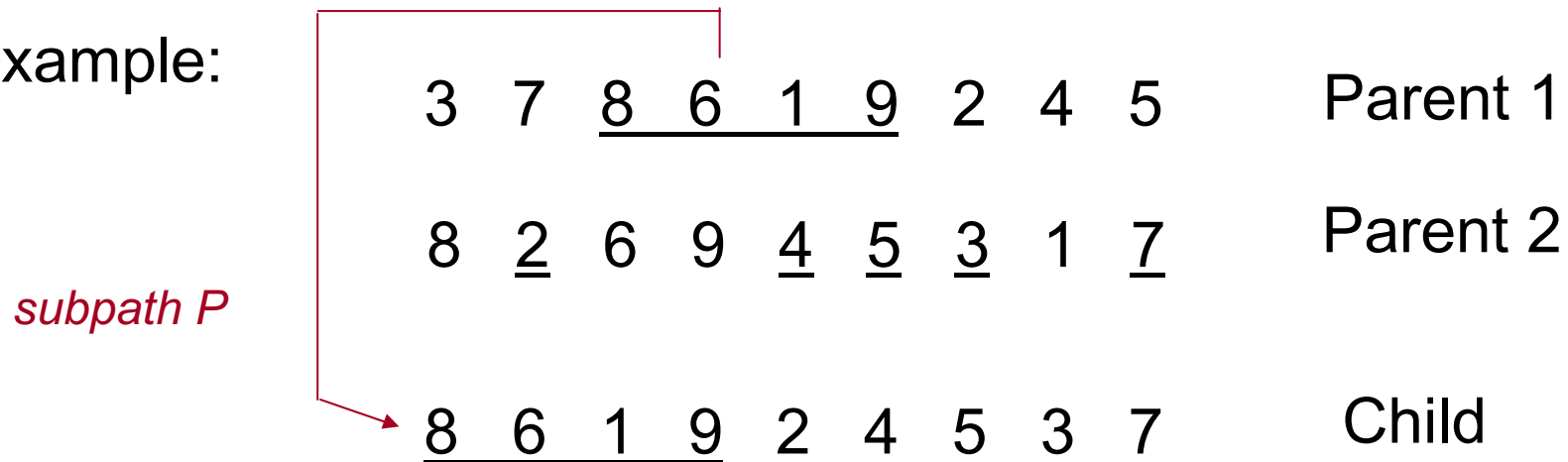
Naïve 1-point
crossover does not
produce a valid route.

1	2	3		4	5
5	3	1		3	1

doesn't work !!!

- Select a random subpath P from parent 1
- Turn the subpath P into a complete tour by visiting the cities not in P in the order they appear in parent 2

Example:

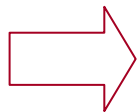


- Can replace an entire population at a time (go from generation k to $k+1$ with no survivors)
 - select $N/2$ pairs of parents
 - create N children, replace all parents
 - polygamy is generally allowed
- Can select two parents at a time
 - create one child
 - eliminate one member of population (weakest?)
- “Elitist” strategy
 - small number of fittest individuals survive unchanged
- “Hall-of-fame”
 - remember best past individuals, but don’t use them for progeny

N = # of members
in population
if steady state

Replacement scheme specifies, how individuals from the parent generation k are chosen to be replaced by children from next generation $k+1$:

- replace all
- replace worst
- replace parent
- replace random
- replace most similar there are others



Academic question: what happens in real life?

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

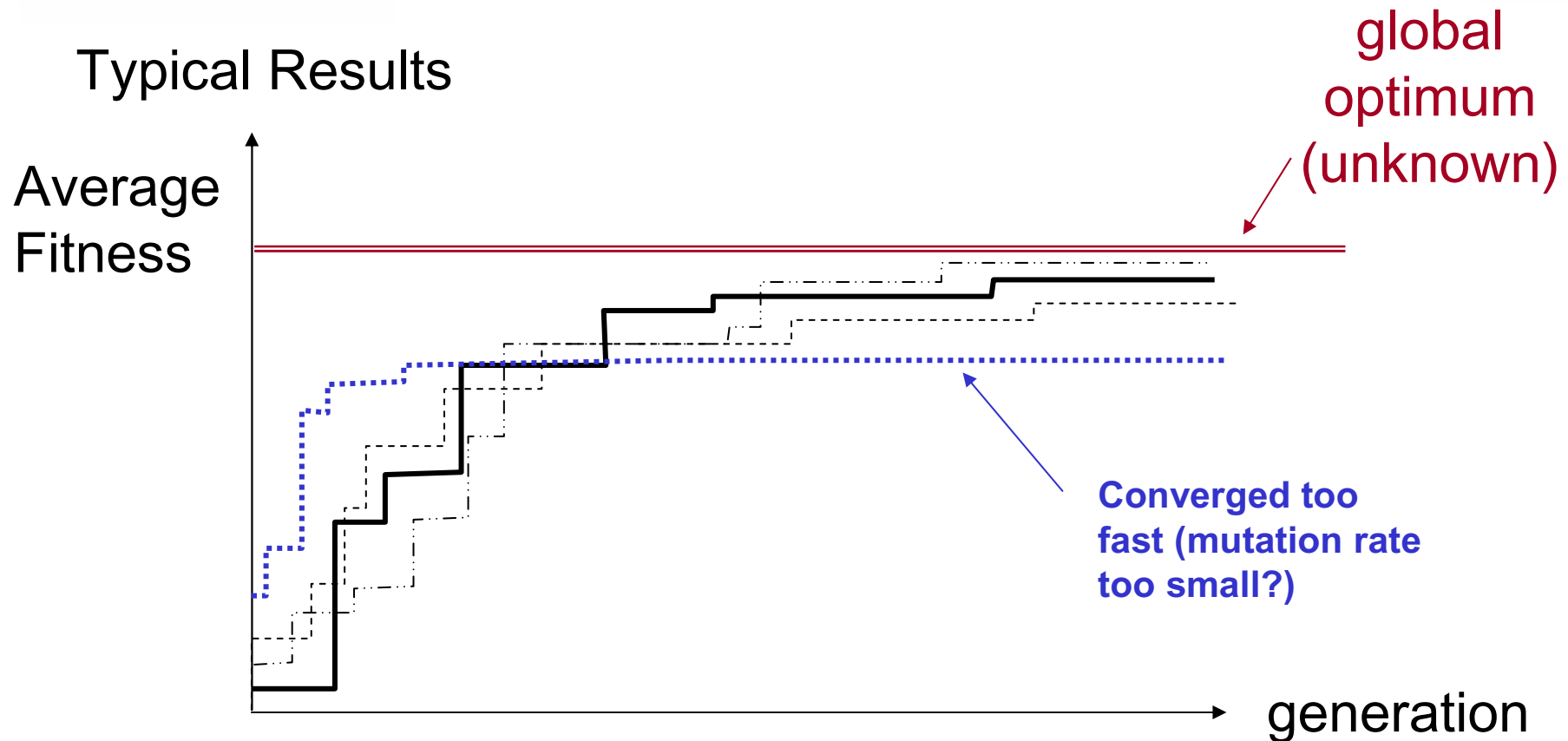
Example:

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

Need to generate $N_{ind} \times L_{ind}$ random numbers from $\{0,1\}$

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

Typical Results



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

Again as for other heuristics there are no clear, obvious termination criteria.

Some options:

- X number of **generations completed** - typically $O(100)$
- **mean deviation** in performance of individuals in the population falls below a threshold $\sigma_J < x$ (genetic diversity has become small)
- **Stagnation** - no or marginal improvement from one generation to the next: $(J_{n+1} - J_n) < X$
- A **particular point** in the search space is encountered

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 parameter set** rather than the parameter set itself
- Gas **do not require derivative information** or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search

Heuristic Methods 2nd part

- Mutation Operator, Schema Theorem
- Speciality GA's
- Particle Swarm Optimization (PSO)
- Tabu Search
- Selection of Optimization Algorithms
- Example Design Applications

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

A.S. Schulz, 15.057 Systems Optimization, Course Notes. Note
a number of charts in this lecture are derived from these class
notes.

T. Baeck, Oxford, N.Y. (1996)
Evolutionary Algorithms in Theory and Practice

A. Zalzala, P.J. Fleming, “Genetic Algorithms in Engineering
Systems” Control Engineering Series 55,
The Institution of Electrical Engineers (IEE), 1997