Genetic Algorithms

Dr. David Fernández Barrero
Dra. Mª Dolores Rodríguez Moreno

Universidad de Alcalá ISG

Objectives

Specific Objectives

- Overview of Genetic Algorithm

Source

- Eiben, A.E. and Smith, J. E. (2003). Introduction to evolutionary computing. Springer

Outline

Introduction (I)

Introduced by Holland in the 70's

- John H. Holland "*Adaptation in Natural and Artificial Systems*", MIT Press
- GA is the most popular EA
- Usually EAs confused with GA

Canonical GA (which is not canonical)

- Fixed length strings
- Binary codification

Representation	Bit strings
Recombination	1-point
Mutation	Bit flip
Parent select	Fitness prop
Survivor select	Generational

Introduction (II)

- The GA is a probabilistic search algorithm that iteratively transforms a set (called a population) of mathematical objects each with an associated fitness value, into a new population of offspring using the Darwinian principle
- Attributed features
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators

Introduction (III)

GAs have common features

- Representation in strings, named chromosomes
- Mutation and recombination
- Usually fixed length

GAs are like a toolbox with customizable components

- Representations, genetic operators, selections mechanism, ...
- These components are interdependent

Rule of thumb: Small genotype changes $\Longrightarrow$ Small phenotype changes

Introduction (III)

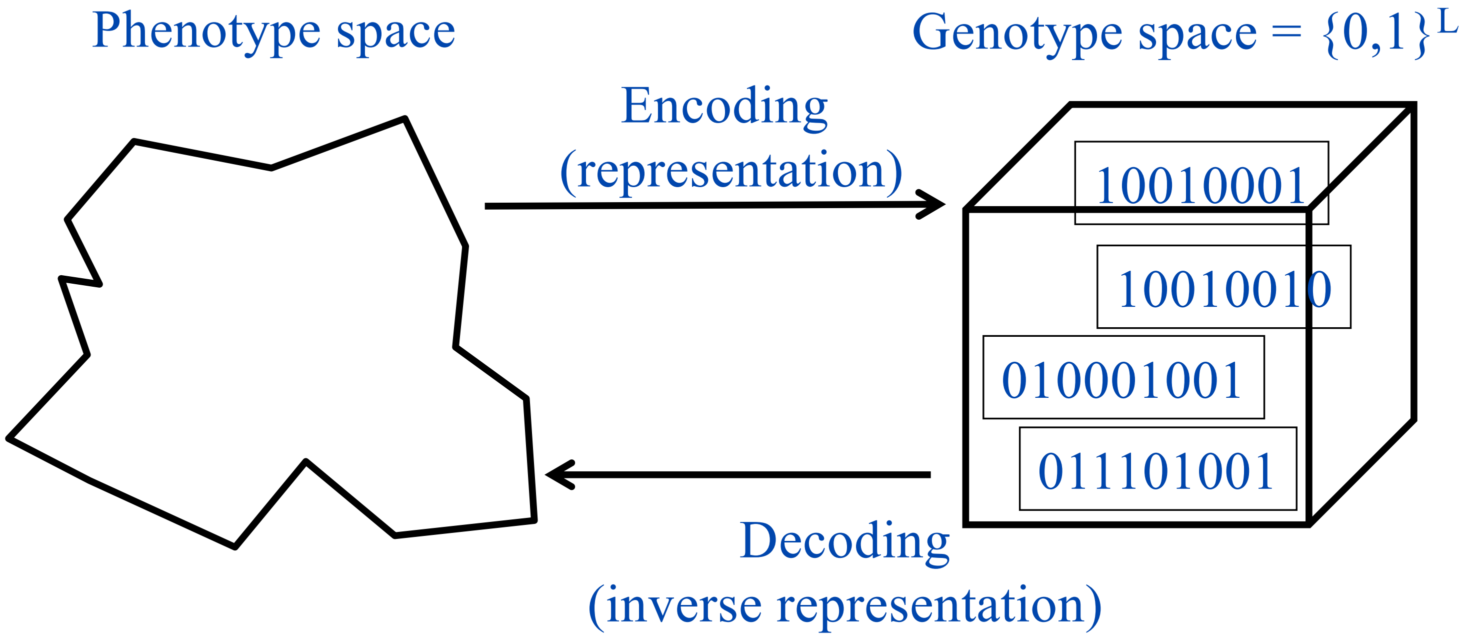
1. Select parents for the mating pool (size of mating pool = population size)
2. Shuffle the mating pool
3. For each consecutive pair apply crossover with probability pc , otherwise copy parents
4. For each offspring apply mutation (bit-flip with probability pm independently for each bit)
5. Replace the whole population with the resulting offspring

Introduction (IV)

Representation	Binary strings
Recombination	N-point or uniform
Mutation	Bitwise bit-flipping with fixed probability
Parent selection	Fitness-Proportionate
Survivor selection	All children replace parents
Speciality	Emphasis on crossover

Outline

Representation



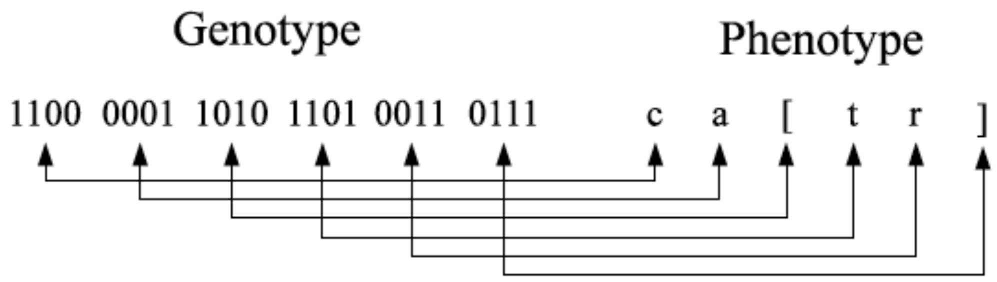
Binary Representation (I)

- The chromosome should in some way contain information about solution which it represents
- The most used way of encoding is a binary string, although depending of the problem one can encode directly integer or real numbers
- The chromosome could look like this:

Chromosome 1	**1101100100110110**
Chromosome 2	1101111000011110

Binary Representation (II)

- Each chromosome has one binary string
- Each bit in this string can represent some characteristic of the solution or the whole string can represent a number
- Often used to codify non-binary information (not recommended)
  - Pure binary codification
  - Gray coding
  - Custom codification



Integer Representation

- Binary representation not always suitable where different genes can take one of a set of values
- Some problems naturally have integer variables, e.g. image processing parameters
- Others take *categorical* values from a fixed set e.g. {blue, green, yellow, pink}

4	3	2	1	0	4	2	3	3

Real-Valued Representation

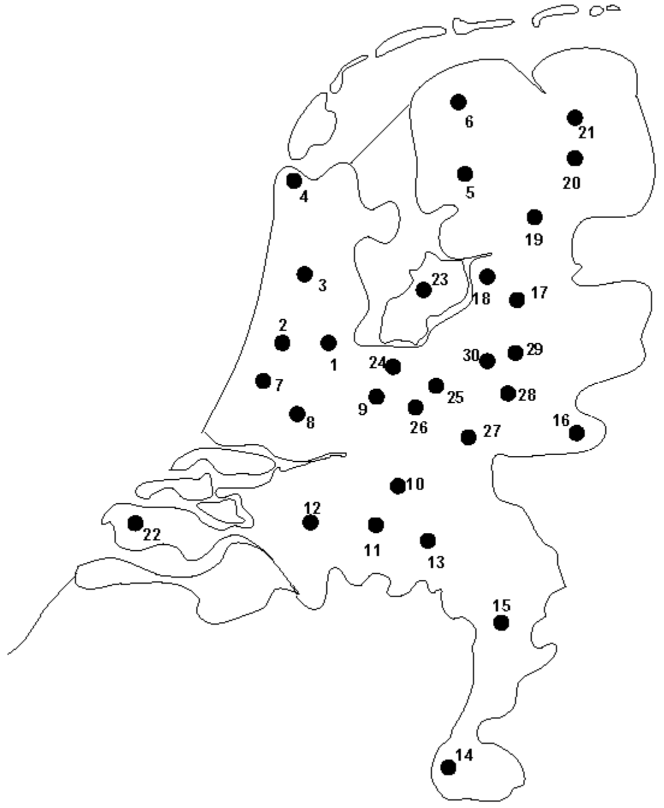
- Often most sensible way to represent a candidate solution is a string of real values
- When? Represent genes from a continuous rather than discrete distribution
- Depends on the computer precision = floating-point numbers

1.1	0.2	3.0	33.2	0.0	-3.2	130.1	88.3	-7.1

Permutation Representation

- Ordering/sequencing problems form a special type
- GA string allows numbers to occur more than once, such sequence of integers will not represent valid permutations
- Task is (or can be solved by) arranging some objects in a certain order
  - Example: sort algorithm: important thing is which elements occur before others (order)
  - Example: Travelling Salesman Problem (TSP) : important thing is which elements occur next to each other (adjacency)
- These problems are generally expressed as a permutation:
  - if there are *n* variables then the representation is as a list of *n* integers, each of which occurs exactly once

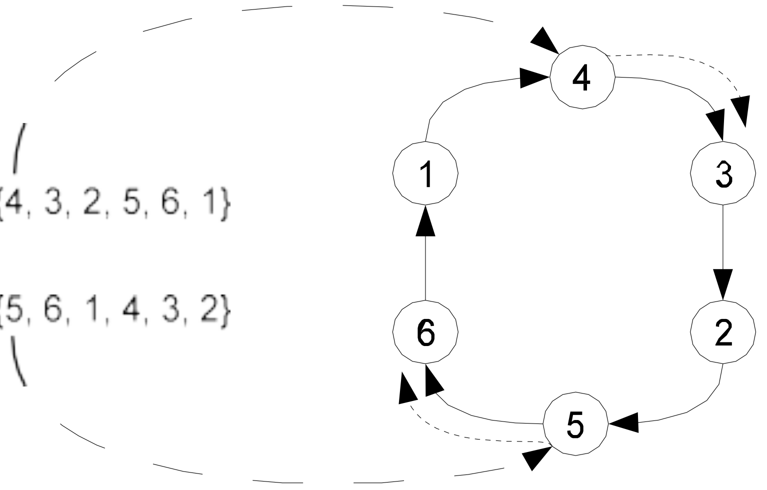
PR: TSP example



- Some problems involve order
- Sequence of integers
- No repeated numbers
- Range of valid numbers
- Special genetic operators

$G_1 = \{4, 3, 2, 5, 6, 1\}$

$G_2 = \{5, 6, 1, 4, 3, 2\}$



Integer codification to solve TSP

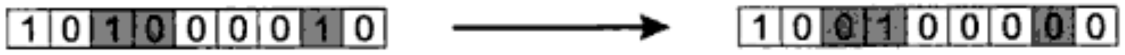
Outline

Mutation

- The generic name given to operators that use one parent and create one child by applying some random change to representation (genotype)
- The form taken depends on the encodings and the associated parameters (mutation rate)

Mutation for BR

- Most common considers each gene separately and allows each bit to flip with a small p_m
- Number of values changed is not fixed, depends on the sequence of random numbers
  - L is the encoding, on average $L.p_m$ values will be changed



  - The p_m value depends on the desired outcome
  - Most binary coded GAs use mutation rates in a range that in average 1 gene per generation and 1 gene per offspring is mutated

Mutation for IR

- 2 forms with user-defined p_m
- Random Resetting
  - Cardinal attributes
  - All genes are equally likely to be chosen
  - With p_m a new value is chosen at random from the set of permissible values in each position
- Creep Mutation
  - Ordinal attributes
  - Add a small (+ or -) value to each gene with p_m

Mutation for FP-R

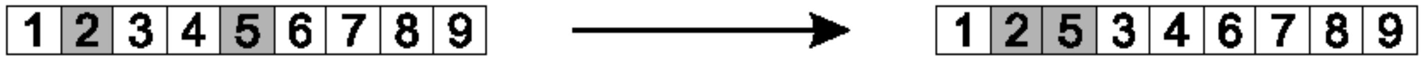
- Change the allele value for each gene randomly within its domain given Lower (L_i) and Upper (U_i) bound
- 2 types
- Uniform Mutation
  - The alleles values are drawn uniformly randomly from $[L_i, U_i]$
- Nonuniform with Fixed Distribution
  - Designed to introduce small changes
  - Add a value from a Gaussian distribution, with mean zero and user-specified standard deviation, and then, curtailing the resulting value to $[L_i, U_i]$ if necessary

Mutation for P-R

- Each gene cannot be considered independent
- Find legal mutations moving alleles around the genome
- Mutation parameter now reflects the probability that some operator is applied once to the whole string, rather than individually in each position
- 4 types
  - Insert Mutation
  - Swap Mutation
  - Inversion Mutation
  - Scamble Mutation

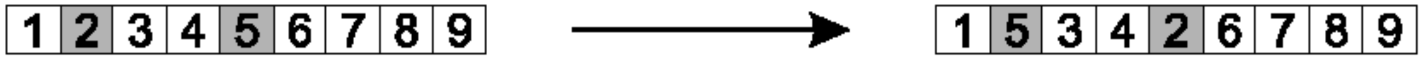
Insert Mutation for PR

- Pick two allele values at random
- Move the second to follow the first, shifting the rest along to accommodate
- Note that this preserves most of the order and the adjacency information



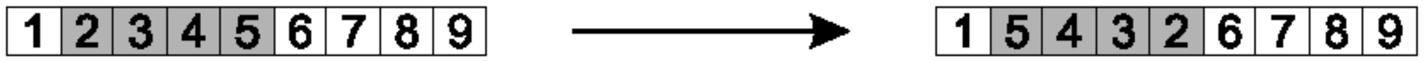
Swap mutation for PR

- Pick two alleles at random and swap their positions
- Preserves most of adjacency information (4 links broken), disrupts order more



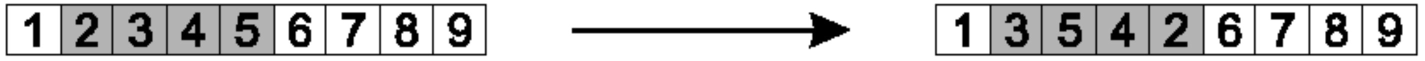
Inversion mutation for PR

- Pick two alleles at random and then invert the substring order between them.
- Preserves most adjacency information (only breaks two links) but disruptive of order information



Scramble mutation for PR

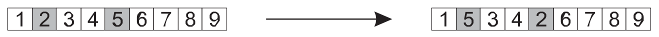
- Pick a subset of genes at random
- Randomly rearrange the alleles in those positions



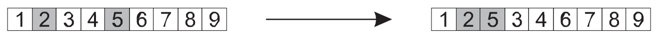
(note subset does not have to be contiguous)

Mutation summary for PR

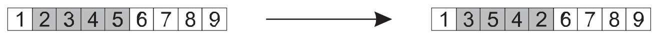
Swap mutation



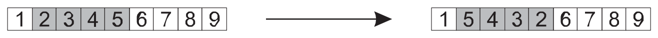
Insert mutation



Scramble mutation



Inversion mutation



Outline

Recombination

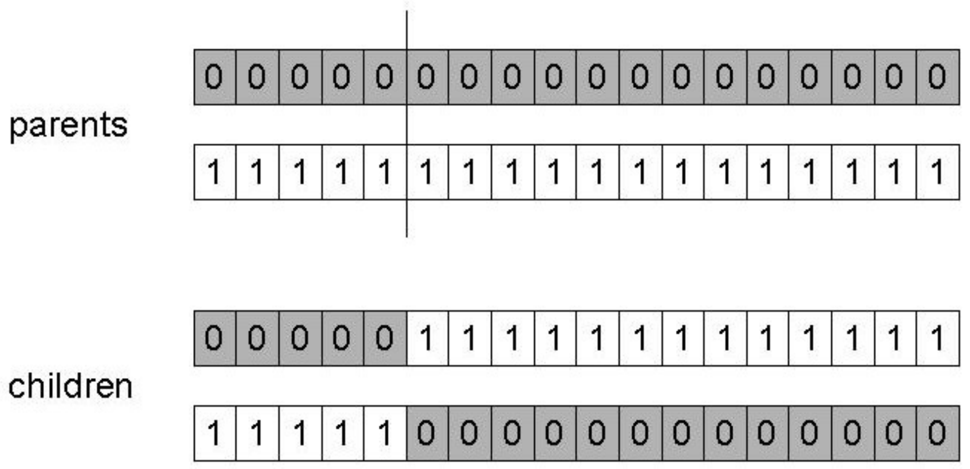
- Is the process of creating a new individual from the information contained within 2 (o +) parents
- Mechanism to create diversity (= crossover)
  - Applied probabilistically with p_c =[0.5, 1]
  - 2 parents selected, a random var. is drawn from [0,1) and compared to p_c
    - If lower: 2 offspring created by recombination or parents
    - Else are created asexually (i.e. copying the parents)
- Determines the chance that a chosen pair of parents undergoes this operator

Crossover for BR &IR

- Start from 2 parents and create 2 children
- 3 types
  - 1-point Crossover
  - N-point Crossover
  - Uniform Crossover

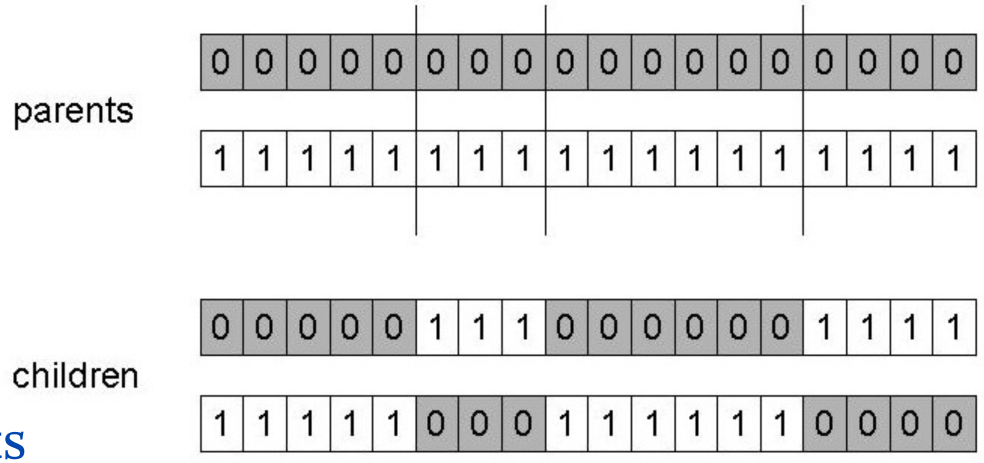
1-point crossover

- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails



n-point crossover

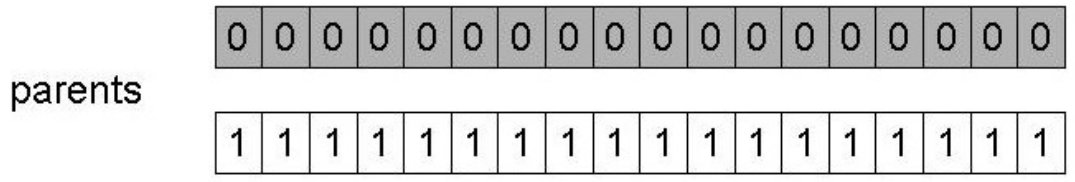
- Choose n random crossover points
- Split along those points



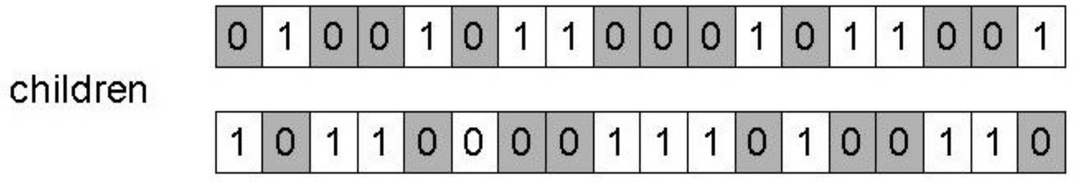
- Glue parts, alternating between parents
- Generalisation of 1 point (still some positional bias)

Uniform crossover

- It works by treating each gene independently and making a random choice as to which parent it should be inherited from
- Assign 'heads' to one parent, 'tails' to the other
- Flip a coin for each gene of the first child

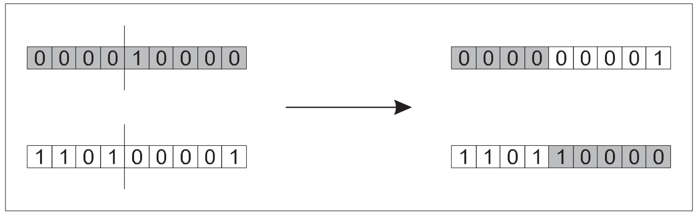


- Make an inverse copy of the gene for the second child

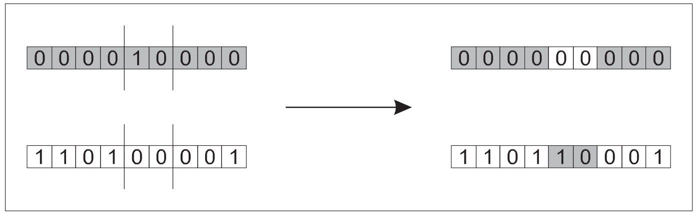


Summary Crossover (BR & IR)

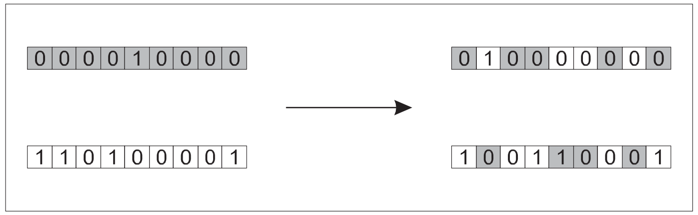
One-point crossover



Two-points crossover



Uniform crossover

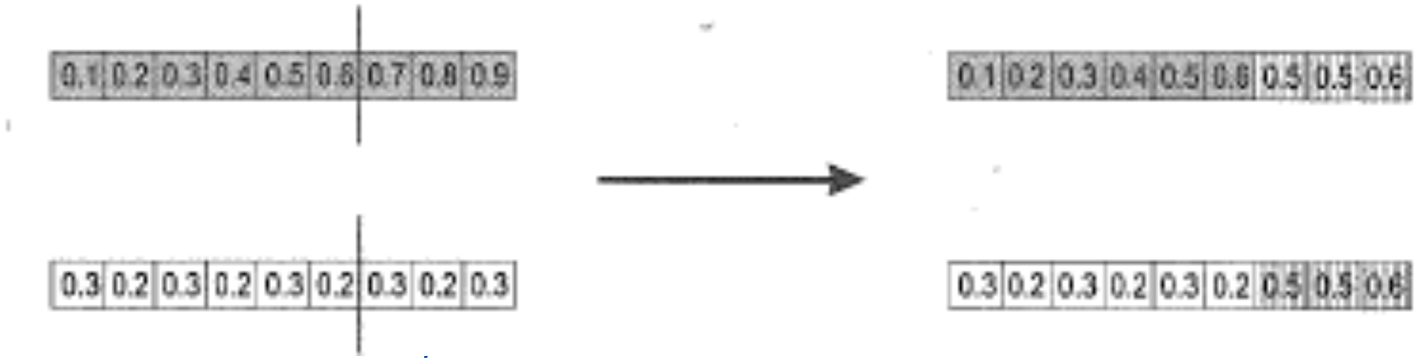


Crossover for FR

1. Use = operators for bit-strings but now split between floats
   - That is, an allele is 1 floating-point value instead of 1 bit (**Discrete Crossover**)
   - Just M can insert new values, since C just combinations of existing floats
   - If z (offspring) and x & y (parents), then the allele value for gene i is $z_i = x_i$ or y_i with the same likelihood
2. In each gene position create a new allele value in the offspring that lies between those of the parents
   - $z_i = \alpha x_i + (1 - \alpha) y_i \quad \alpha \in [0, 1]$
   - Create new gene material (**Arithmetic Crossover**)
     a) Simple Recombination
     b) Single Recombination
     c) Whole Recombination

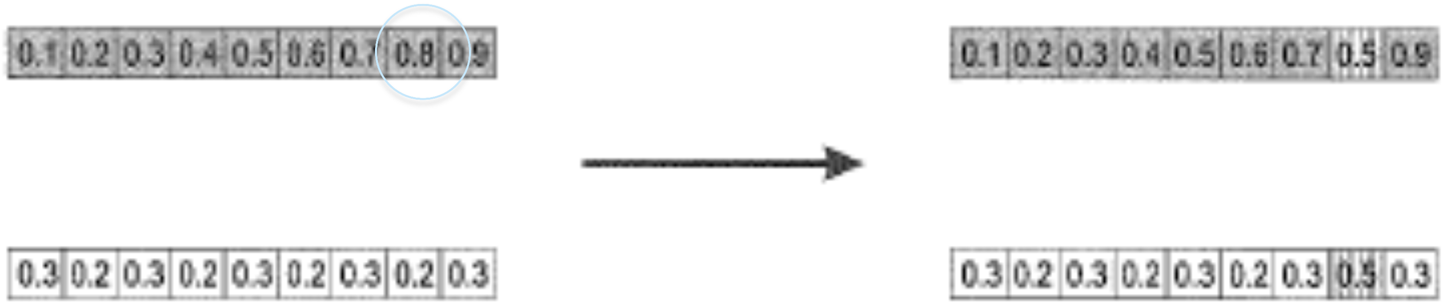
Simple Arithmetic R (FR)

- Pick a recombination point k
- For child 1, take the first k floats of parent 1 and put them into the child
- The rest is the arithmetic average of parent 1 & 2



Single Arithmetic R (FR)

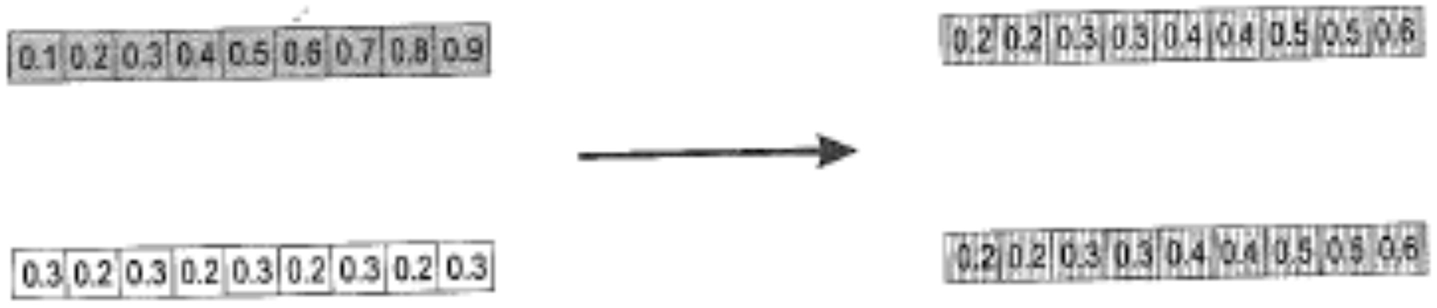
- Pick a random allele k
- At that position, take the arithmetic average of the two parents
- Child 1: $\langle x_1, \ldots, x_{k-1}, \alpha \cdot y_k + (1-\alpha) \cdot x_k, x_{k+1}, \ldots, x_n \rangle.$
- Child 2 with x and y reversed



Whole Arithmetic R (FR)

- Takes the weighted sum of 2 parental alleles for each gene

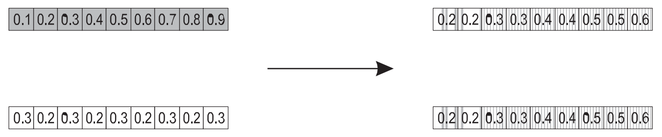
$$\text{Child } 1 = \alpha \cdot \bar{x} + (1-\alpha)\cdot \bar{y}, \qquad \text{Child } 2 = \alpha \cdot \bar{y} + (1-\alpha)\cdot \bar{x}.$$



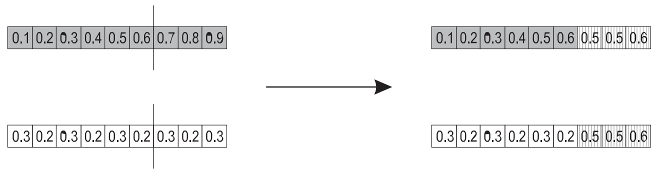
$\alpha = 1/2$

Summary R for FR

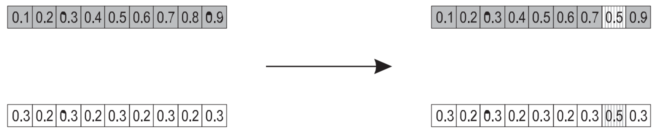
Whole arithmetic recombination
(All genes are included)



Simple arithmetic recombination
(Similar to one-point crossover)

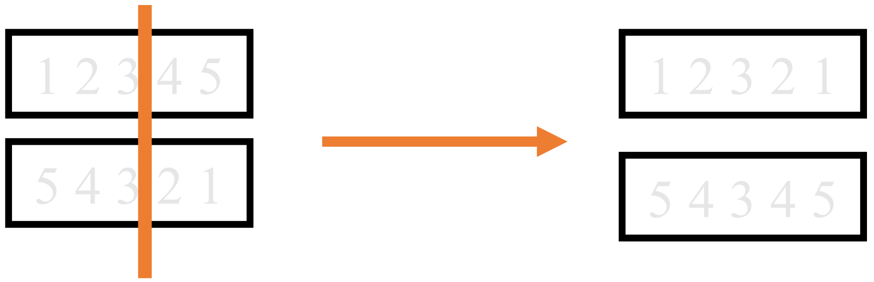


Single arithmetic recombination
(Similar to uniform crossover)



Crossover operators for P

- "Normal" crossover operators will often lead to inadmissible solutions



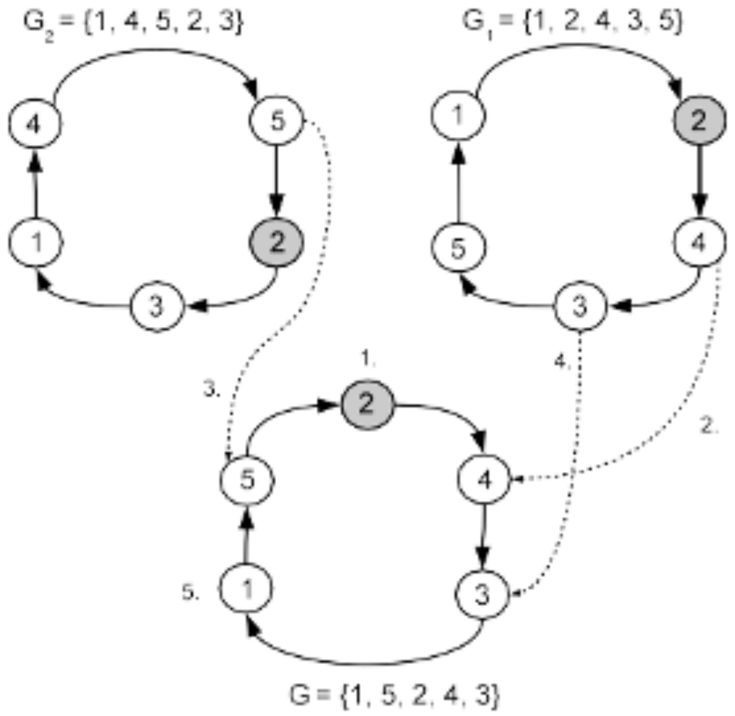
- Many specialised operators have been devised which focus on combining order or adjacency information from the two parents

Crossover operators for P

- Idea is to preserve relative order that elements occur
- Informal procedure:
  1. Choose an arbitrary part from the first parent
  2. Copy this part to the first child
  3. Copy the numbers that are not in the first part, to the first child:
     - starting right from cut point of the copied part,
     - using the **order** of the second parent
     - and wrapping around at the end
  4. Analogous for the second child, with parent roles reversed

Crossover operators for P

- 4 types:
  - Partially Mapped Crossover
  - Edge Crossover
  - Order Crossover
  - Cycle Crossover



Crossover OR mutation?

- Decade long debate: which one is better / necessary / main-background

- Answer (at least, rather wide agreement):
  - it depends on the problem, but
  - in general, it is good to have both
  - both have a role
  - mutation-only-EA is possible, xover-only-EA would not work

Outline

Population Models

- SGA uses a Generational model (GGA):
  - each individual survives for exactly one generation
  - the entire set of parents is replaced by the offspring
- At the other end of the scale are Steady-State models (SSGA):
  - part of the population is replaced
- Generation Gap
  - the proportion of the population replaced
  - Values: 1.0 for GGA, 1/pop_size for SSGA

Outline

- Introduction
- Representation of individuals
- Mutation
- Recombination
- Population Models
- **Parent Selection**
- Survivor Selection
- Conclusions

Parent Selection (I)

- Selection can occur in two places:
  - Selection from current generation to take part in mating (parent selection)
  - Selection from parents + offspring to go into next generation (survivor selection)
- Selection operators work on whole individual
  - i.e. they are representation-independent
- Distinction between selection
  - operators: define selection probabilities
  - algorithms: define how probabilities are implemented

Parent Selection (II)

- 4 methods
  - Fitness Proportionate Selection
  - Raking Selection
  - Selection Probabilities
  - Tournament Selection

Fitness-Proportionate Selection

- Problems include
  - One highly fit member can rapidly take over if rest of population is much less fit: Premature Convergence
  - At end of runs when fitnesses are similar, lose selection pressure

Rank – Based Selection

- Attempt to remove problems of FPS by basing selection probabilities on *relative* rather than *absolute* fitness
- Rank population according to fitness and then base selection probabilities on rank where fittest has rank μ (= pop.size) and worst rank 1
- This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time

Tournament Selection

- All methods above rely on global population statistics
  - Could be a bottleneck esp. on parallel machines
  - Relies on presence of external fitness function which might not exist: e.g. evolving game players
- Informal Procedure:
  - Pick *k* members at random then select the best of these
  - Repeat to select more individuals

Outline

- Introduction
- Representation of individuals
- Mutation
- Recombination
- Population Models
- Parent Selection
- **Survivor Selection**
- Conclusions

Survivor Selection

- Survivor selection can be divided into two approaches:
  - Age-Based Selection
    - e.g. SGA
    - In SSGA can implement as "delete-random" (not recommended) or as first-in-first-out (a.k.a. delete-oldest)
  - Fitness-Based Selection
    - Using one of the methods above

Two Special Cases

- Elitism
  - Widely used in both population models (GGA, SSGA)
  - Always keep at least one copy of the fittest solution so far
- GENITOR:
  - "delete-worst"
  - Rapid takeover : use with large populations or "no duplicates" policy

Outline

Conclusions

- GA is a search heuristic that mimics the process of natural evolution
- It uses techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover
- The heuristic is used to generate useful solutions to optimization and search problems

Outline

An example after Goldberg '89 (1)

- Simple problem: max x^2 over {0,1,…,31}
- GA approach:
  - Representation: binary code, e.g. 01101 $\leftrightarrow$ 13
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel selection
  - Random initialisation
- We show one generational cycle done by hand

x^2 example: selection

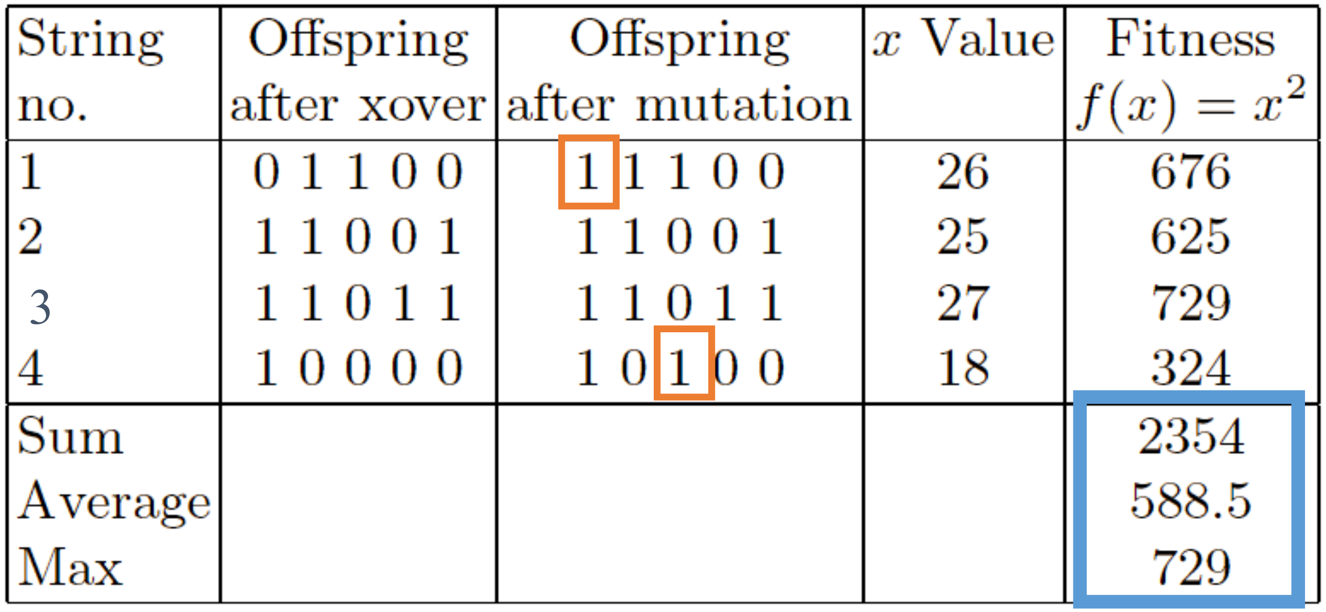
String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

Probability to be chosen as father ($f_i/\Sigma f_i$)

Numb of copies (f_i/Average f)

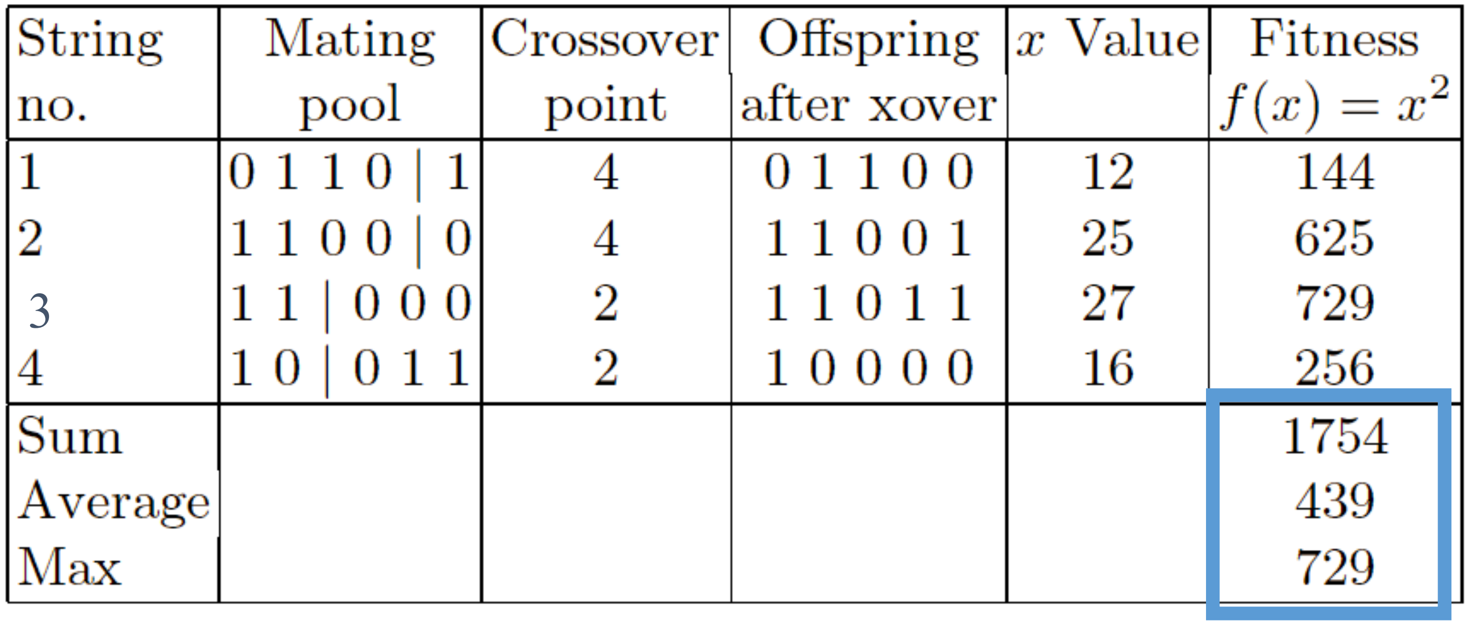
Numb of individual next generation

X^2 example: mutation



String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	$\boxed{1}$ 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
3	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 $\boxed{1}$ 0 0	18	324
Sum				2354
Average				588.5
Max				729

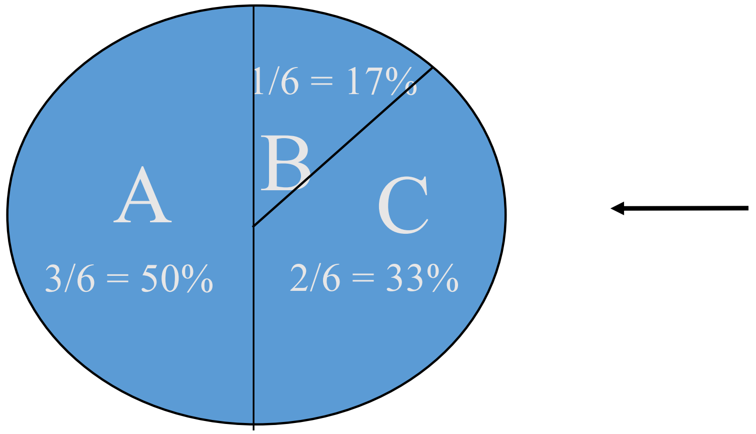
x^2 example: crossover



String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
3	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

GA operators: selection

- Main idea: better individuals get higher chance
    - Chances proportional to fitness
    - Implementation: roulette wheel technique
        - Assign to each individual a part of the roulette wheel
        - Spin the wheel n times to select n individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2