Genetic Algorithms

Chapter 3-1

Source: Eiben & Smith

Edited by: Hosein Alizadeh
http://webpages.iust.ac.ir/halizadeh/
Computer Engineering Department, Iran University of Science & Technology

GA Quick Overview

- Developed: USA in the 1970's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete optimization
- 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

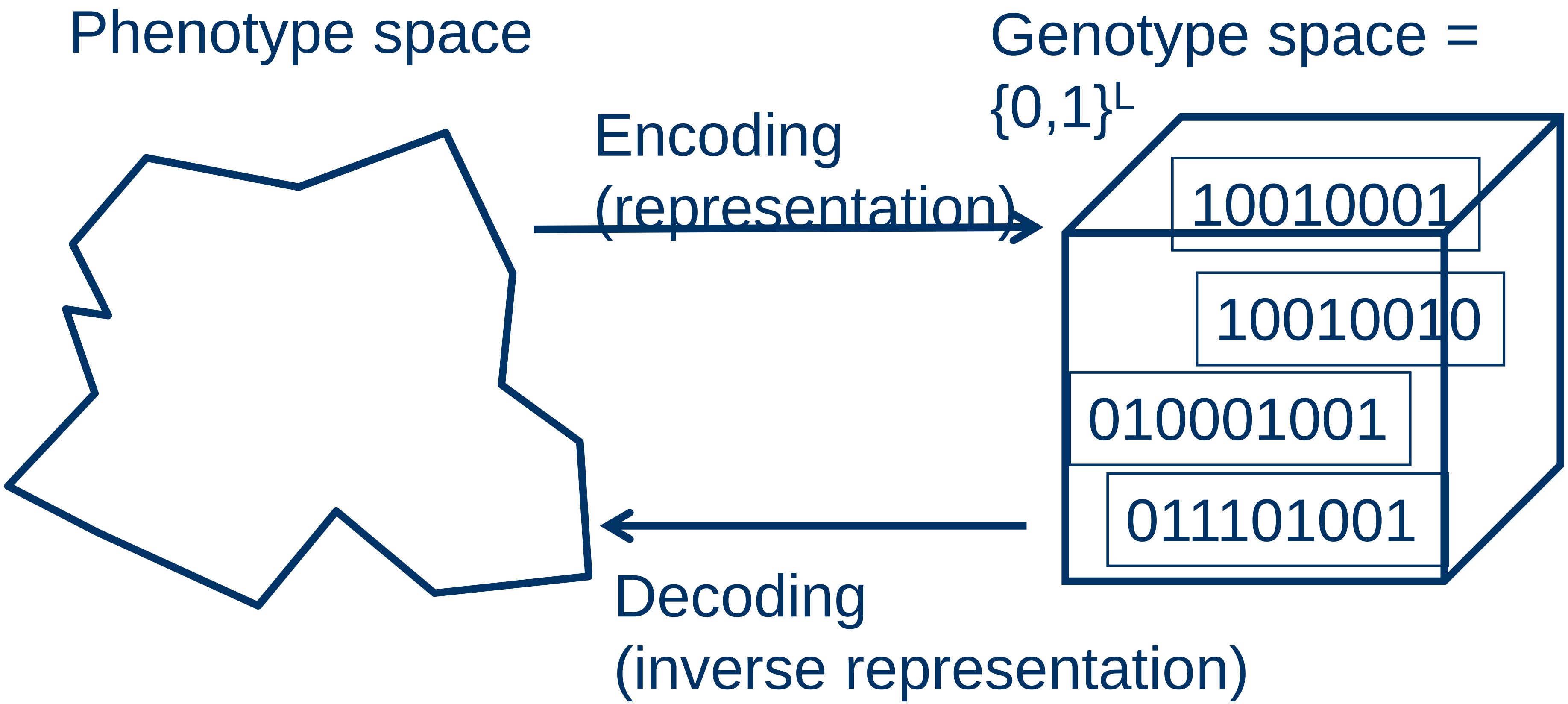
Genetic algorithms

- Holland's original GA: **Simple GA** (SGA)
- Other GAs use different:
  - Representations
  - Mutations
  - Crossovers
  - Selection mechanisms

SGA technical summary tableau

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

Representation

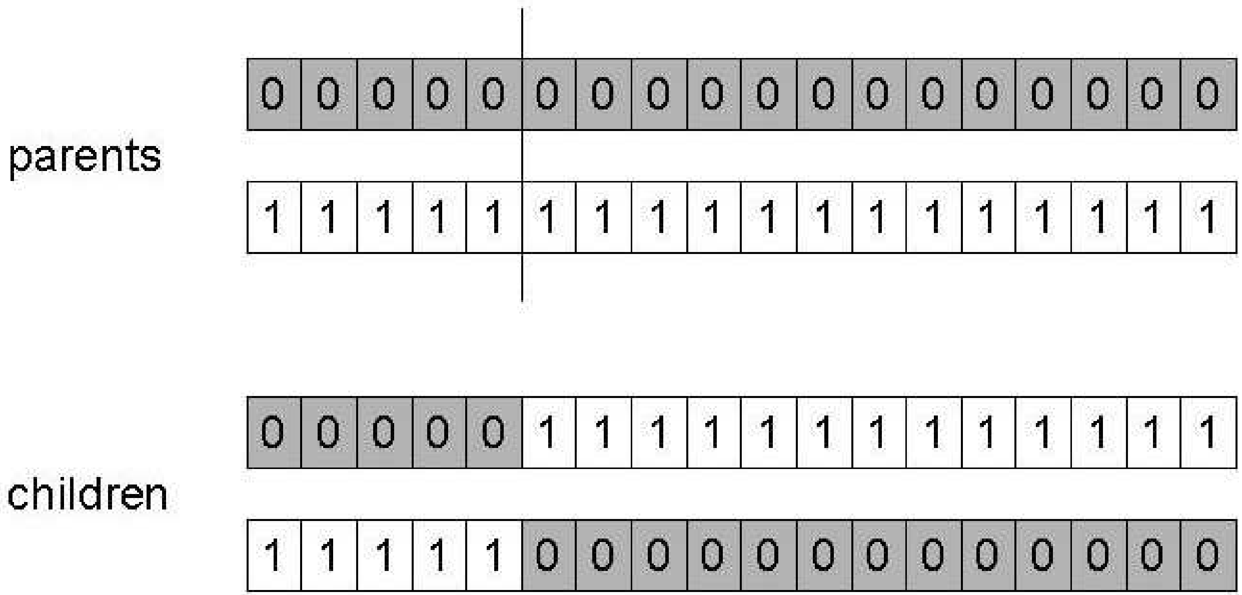


SGA reproduction cycle

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

SGA operators: 1-point crossover

- Choose a random point
- Split parents
- Create children by exchanging tails
- P_c typically in range (0.6, 0.9)



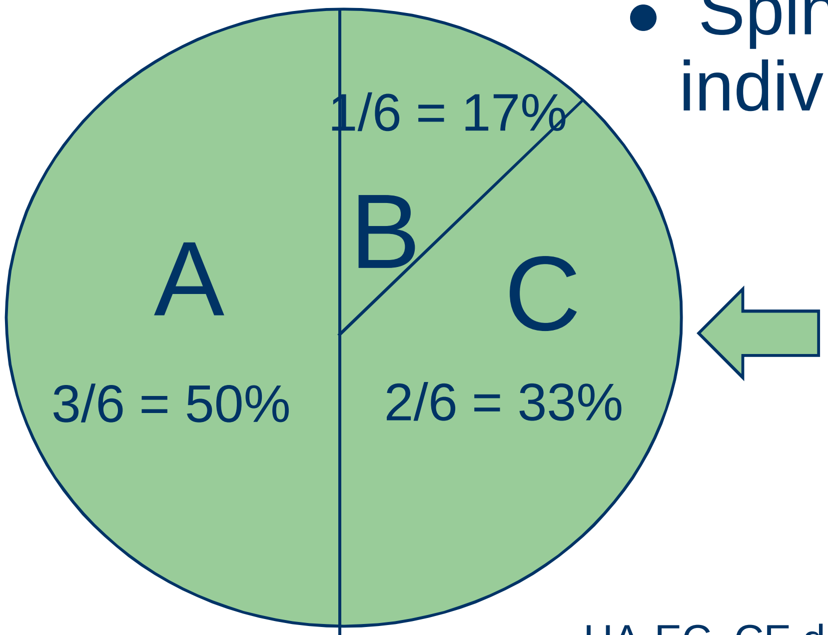
SGA operators: mutation

- Alter each gene independently with p_m
- p_m : mutation rate
  - Typically between 1/pop_size and 1/ chromosome_length

parent	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
child	0	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	1

SGA operators: Selection

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



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

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

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
2	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

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
2	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

SGA

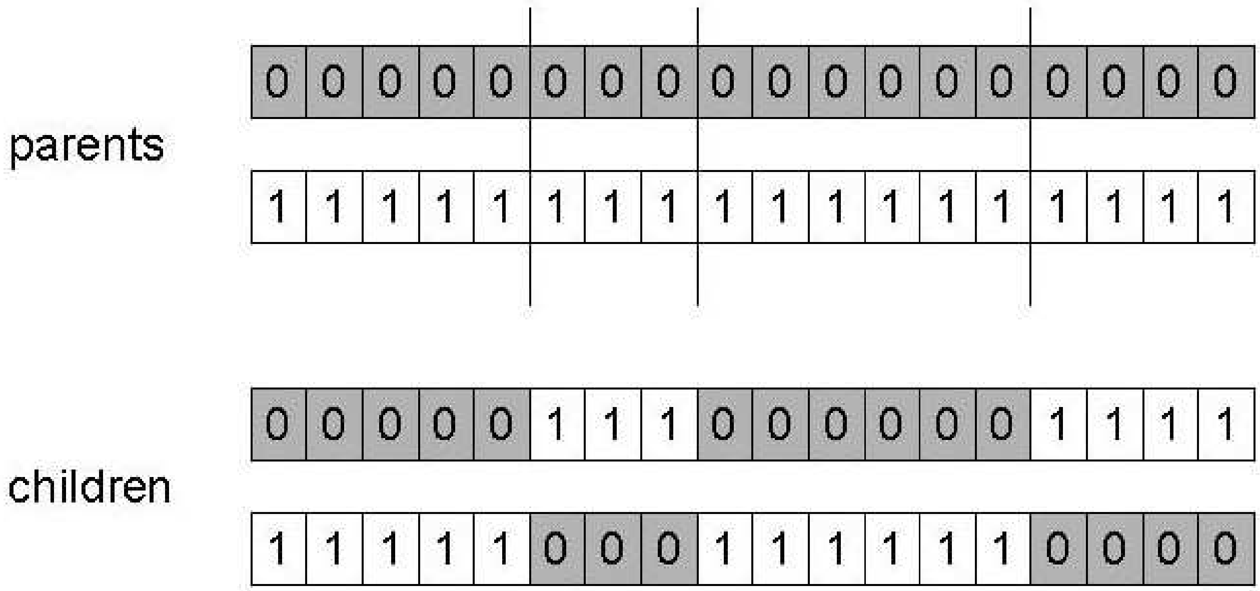
- Subject of many studies
- Shortcomings:
  - Representation: too restrictive
  - Mutation & crossovers: only applicable for bit-string & integer representations
  - Selection mechanism: converging populations with close fitness values
  - Next Generation model (step 5 in SGA): can be improved with better survivor selection

Alternative Crossover Operators

- Performance with 1 Point Crossover depends on the **order** of genes
- *Positional Bias*
  - more likely to keep together genes that are **near** each other
  - Can never keep together genes from **opposite ends**

n-point crossover

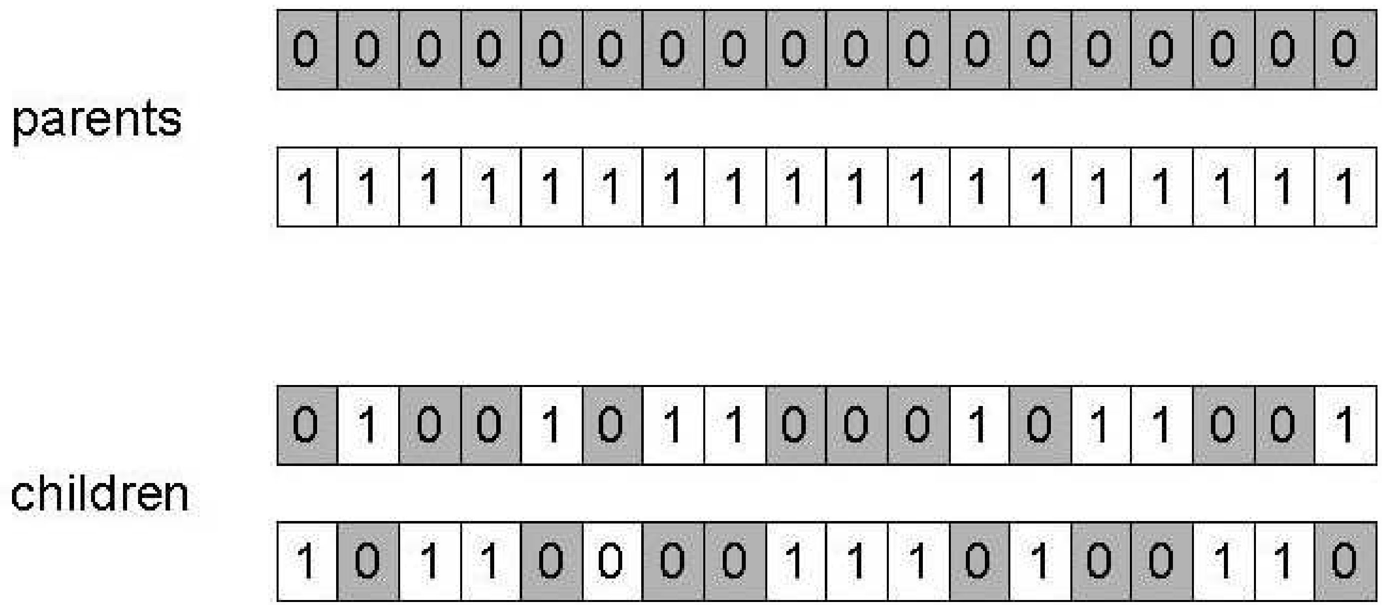
- Choose n random crossover points
- Split along those points
- Exchange parts
- Generalisation of 1 point (still some positional bias)



Uniform crossover

- Assign 'heads' to one parent, 'tails' to the other
- Flip a coin for each gene

- Inheritance is independent of position (No Positional Bias)



Crossover OR mutation?

- which one is better / necessary?

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

Crossover OR mutation? (cont'd)

Exploration: Discovering promising areas in the search space (gaining information)

Exploitation: Optimizing within a promising area (using information)

- Crossover is explorative
  - makes a *big* jump to an area somewhere "between" parent areas
- Mutation is exploitative
  - creates random *small* diversions, staying near the parent

Crossover OR mutation? (cont'd)

- crossover only combine information from two parents
- mutation only introduce new information
- Crossover does not change the allele frequencies of the population (thought experiment: 50% 0's on first bit in the population, ?% after performing *n* crossovers)
- Often need a 'lucky' mutation to hit the optimum

Integer representations

- Problems with integer variables
  - image processing parameters
  - *categorical* values {blue, green, yellow, pink}
- N-point / uniform crossover operators work
- Extend bit-flipping mutation to make
  - "creep" i.e. more likely to move to similar value
  - Random choice (esp. categorical variables)

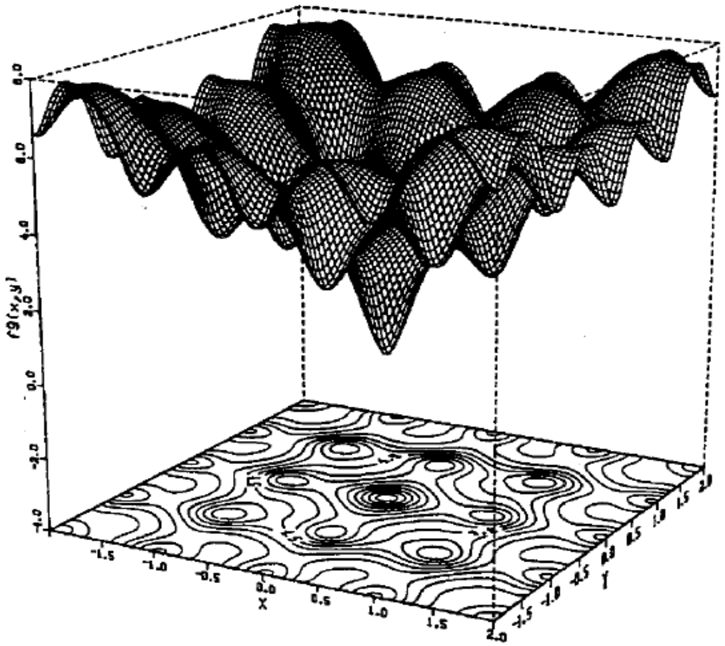
Real valued problems

- Many problems occur as real valued problems
  - continuous parameter Optimization $f : \Re^n \rightarrow \Re$
- Ackley's function

$$f(\overline{x}) = -c_1 \cdot exp\left(-c_2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$$

$$-exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} cos(c_3 \cdot x_i)\right) + c_1 + 1$$

$$c_1 = 20,\ c_2 = 0.2,\ c_3 = 2\pi$$



Single arithmetic crossover

- Parents: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
- Pick a single gene (k) at random,
- child$_1$ is: $\left\langle x_1, \ldots, x_k, \alpha \cdot y_k + (1-\alpha) \cdot x_k, \ldots, x_n \right\rangle$
- reverse for other child. e.g. with $\alpha = 0.5$

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.2	0.3

$\longrightarrow$

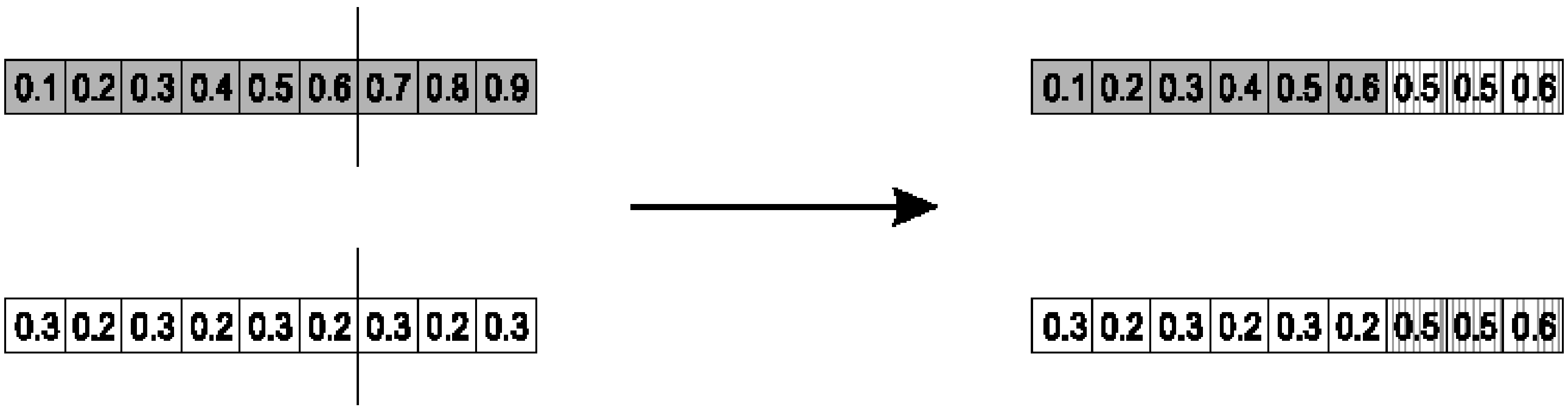
0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.5	0.9
0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.5	0.3

Simple arithmetic crossover

- Parents: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
- Pick random gene *(k)* after this point mix values
- child$_1$ is:

$$\left\langle x_1,\ldots,x_k,\alpha\cdot y_{k+1}+(1-\alpha)\cdot x_{k+1},\ldots,\alpha\cdot y_n+(1-\alpha)\cdot x_n\right\rangle$$

- reverse for other child. e.g. with $\alpha = 0.5$



Whole arithmetic crossover

- Most commonly used
- Parents: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
- $child_1$ is:

$$a \cdot \bar{x} + (1 - a) \cdot \bar{y}$$

- reverse for other child. e.g. with $\alpha = 0.5$

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.3	0.2	0.3	0.2	0.3	0.2	0.3	0.2	0.3

0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5	0.6
0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5	0.6