

Genetic Algorithms

1

- Genetic algorithms (GA's) are a technique to solve problems which need optimization
- GA's are a subclass of **Evolutionary Computing**
- GA's are based on Darwin's theory of evolution
- History of GA's
 - Evolutionary computing evolved in the 1960's.
 - GA's were created by John Holland in the mid-70's.

2

- Genetic information is stored in the **chromosomes**
- Each chromosome is build of **DNA**
- Chromosomes in humans form pairs
- There are 23 pairs
- The chromosome is divided in parts: **genes**
- Genes code for properties
- The possibilities of the genes for one property is called: **allele**
- Every gene has an unique position on the chromosome: **locus**

3

Simple Genetic algorithm

- Holland's original GA is now known as the Simple Genetic Algorithm (SGA)
- Other GAs use different:
 - Representations
 - Mutations
 - Crossovers
 - Selection mechanisms

4

SGA technical summary

Representation	Binary strings
Reproduction	N-Point Crossover, Uniform Crossover
Mutation	Bitwise bit-flipping with fixed probability
Parent selection	Fitness-Proportionate
Survivor selection	All children replace parents
Speciality	Emphasis on crossover

5

Biological Terminology

- gene
 - functional entity that codes for a specific feature e.g. eye color
 - set of possible alleles
- allele
 - value of a gene e.g. blue, green, brown
 - codes for a specific variation of the gene/feature
- locus
 - position of a gene on the chromosome
- genome
 - set of all genes that define a species
 - the genome of a specific individual is called genotype
 - the genome of a living organism is composed of several chromosomes
- population
 - set of competing genomes/individuals

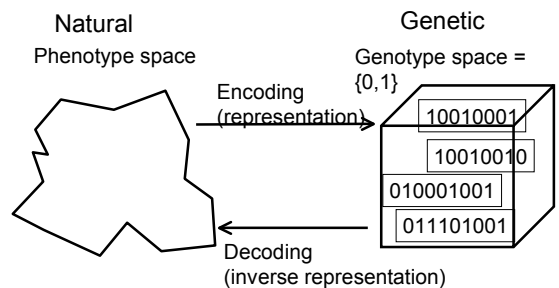
6

Genotype versus Phenotype

- genotype
 - blue print that contains the information to construct an organism e.g. human DNA
 - genetic operators such as mutation and recombination modify the genotype during reproduction
- phenotype
 - physical make-up of an organism
 - selection operates on phenotypes
(Darwin's principle : "survival of the fittest")

7

Representation



8

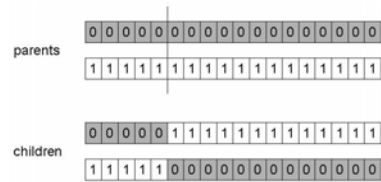
SGA Reproduction Cycle

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 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 resulting offspring

9

SGA operators: 1-point crossover

- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails
- P_c typically in range (0.6, 0.9)



10

SGA operators: Mutation

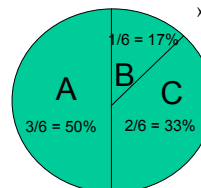
- Alter each gene independently with a probability p_m
- p_m is called the mutation rate
 - Typically between $1/\text{pop_size}$ and $1/\text{chromosome_length}$



11

SGA 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

12

An example: find maximum value of $f(x)$

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

13

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

Expected count = $f_i / \text{average of } f_i$ for $i=1,2,3,4$
 Actual count from Roulette Wheel

14

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

Parents string no. (1,2)
 Parents string no. (3,4)

15

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	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 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

16

The simple GA

- Has been subject of many (early) studies
 - still often used as benchmark for novel GAs
- Shows many shortcomings, e.g.
 - Representation is too restrictive
 - Mutation & crossovers only applicable for bit-string & integer representations
 - Selection mechanism sensitive for converging populations with close fitness values
 - Generational population model (step 5 in SGA repr. cycle) can be improved with explicit survivor selection

17

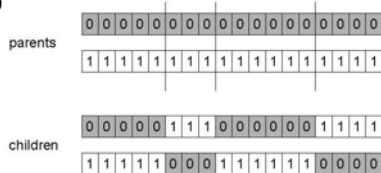
Alternative Crossover Operators

- Performance with 1 Point Crossover depends on the order that variables occur in the representation
 - more likely to keep together genes that are near each other
 - Can never keep together genes from opposite ends of string
 - This is known as *Positional Bias*
 - Can be exploited if we know about the structure of our problem, but this is not usually the case

18

n-point crossover

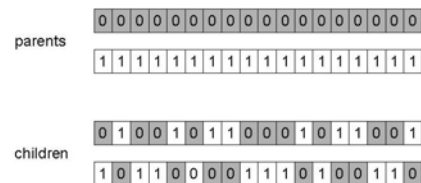
- Choose n random crossover points
- Split along those points
- Glue parts, alternating between parents
- Generalization of 1 point (still some positional bias)



19

Uniform crossover

- 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
- Inheritance is independent of position



20

Crossover OR mutation?

- Decade long debate: which one is better or necessary?
- Answer (at least, rather wide agreement):
 - it depends on the problem, but
 - in general, it is good to have both
 - both have role
 - mutation-only-EA is possible, crossover-only-EA would not work

21

Crossover OR mutation? (cont'd)

Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem

Exploitation: Optimising within a promising area, i.e., using information

There is co-operation AND competition between them

- Crossover is explorative, it makes a *big* jump to an area somewhere “in between” two (parent) areas
- Mutation is exploitative, it creates random *small* diversions, thereby staying near (in the area of) the parent

22

Crossover OR mutation? (cont'd)

- Only crossover can combine information from two parents
- Only mutation can introduce new information (alleles)
- Crossover does not change the allele frequencies of the population
- To hit the optimum we often need a ‘lucky’ mutation

23

Other representations

- Gray coding of integers (still binary chromosomes)
 - Gray coding is a mapping that means that small changes in the genotype cause small changes in the phenotype (unlike binary coding). “Smoother” genotype-phenotype mapping makes life easier for the GA

Nowadays it is generally accepted that it is better to encode numerical variables directly as

- Integers
- Floating point variables

24

Integer representations

- 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}
- 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)

25

Real valued problems

- Many problems occur as real valued problems, e.g. continuous parameter optimization $f: \mathcal{R}^n \rightarrow \mathcal{R}$

26

Mapping real values on bit strings

$z \in [x,y] \subseteq \mathcal{R}$ represented by $\{a_1, \dots, a_L\} \in \{0,1\}^L$

- $[x,y] \rightarrow \{0,1\}^L$ must be invertible (one phenotype per genotype)
- $\Gamma: \{0,1\}^L \rightarrow [x,y]$ defines the representation

$$\Gamma(a_1, \dots, a_L) = x + \frac{y-x}{2^L-1} \cdot \left(\sum_{j=0}^{L-1} a_{L-j} \cdot 2^j \right) \in [x,y]$$

- Only 2^L values out of infinite are represented
- L determines possible maximum precision of solution
- High precision \rightarrow long chromosomes (slow evolution)

27

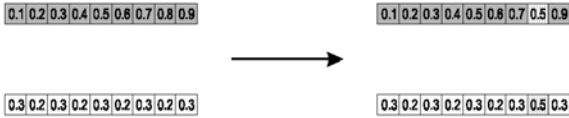
Crossover operators for real valued GAs

- Discrete:
 - each allele value in offspring z comes from one of its parents (x,y) with equal probability: $z_i = x_i$ or y_i
 - Could use n-point or uniform
- Intermediate
 - exploits idea of creating children “between” parents (hence a.k.a. *arithmetic* recombination)
 - $z_i = \alpha x_i + (1 - \alpha) y_i$ where $\alpha: 0 \leq \alpha \leq 1$.
 - The parameter α can be:
 - constant: uniform arithmetical crossover
 - variable (e.g. depend on the age of the population)
 - picked at random every time

28

Single arithmetic crossover

- Parents: $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_n \rangle$
- Pick a single gene (k) at random,
- child₁ is: $\langle x_1, \dots, x_k, \alpha \cdot y_k + (1 - \alpha) \cdot x_k, \dots, x_n \rangle$
- reverse for other child. e.g. with $\alpha = 0.5$



29

Simple arithmetic crossover

- Parents: $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_n \rangle$
- Pick random gene (k) after this point mix values
- child₁ is: $\langle x_1, \dots, x_k, \alpha \cdot y_{k+1} + (1 - \alpha) \cdot x_{k+1}, \dots, \alpha \cdot y_n + (1 - \alpha) \cdot x_n \rangle$
- reverse for other child. e.g. with $\alpha = 0.5$



30

Whole arithmetic crossover

- Most commonly used
- Parents: $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_n \rangle$
- child₁ is: $a \cdot \bar{x} + (1 - a) \cdot \bar{y}$
- reverse for other child. e.g. with $\alpha = 0.5$



31

Permutation Representations

- Ordering/sequencing problems form a special type
- 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

32

Permutation representation: TSP example

- Problem:
 - Given n cities
 - Find a complete tour with minimal length
- Encoding:
 - Label the cities $1, 2, \dots, n$
 - One complete tour is one permutation (e.g. for $n=4$, $[1,2,3,4]$, $[3,4,2,1]$ are OK)
- Search space is BIG:
for 30 cities there are $30! \approx 10^{32}$ possible tours

33

Mutation operators for permutations

- Normal mutation operators lead to inadmissible solutions
 - e.g. bit-wise mutation : let gene i have value j
 - changing to some other value k would mean that k occurred twice and j no longer occurred
- Therefore must change at least two values
- Mutation parameter now reflects the probability that some operator is applied once to the whole string, rather than individually in each position

34

Insert Mutation for permutations

- 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

1 2 3 4 5 6 7 8 9 → 1 2 5 3 4 6 7 8 9

35

Swap mutation for permutations

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

1 2 3 4 5 6 7 8 9 → 1 5 3 4 2 6 7 8 9

36

Inversion mutation for permutations

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

1 2 3 4 5 6 7 8 9 → 1 5 4 3 2 6 7 8 9

37

Scramble mutation for permutations

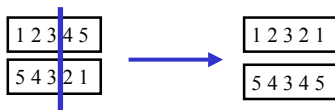
- Pick a subset of genes at random
- Randomly rearrange the alleles in those positions

1 2 3 4 5 6 7 8 9 → 1 3 5 4 2 6 7 8 9

38

Crossover operators for permutations

- “Normal” crossover operators will often lead to inadmissible solutions



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

39

Order 1 crossover

- 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

40

Order 1 crossover example

- Copy randomly selected set from first parent

1 2 3 4 5 6 7 8 9



4 5 6 7

9 3 7 8 2 6 5 1 4

- Copy rest from second parent in order 1,9,3,8,2

1 2 3 4 5 6 7 8 9



3 8 2 4 5 6 7 1 9

9 3 7 8 2 6 5 1 4

42

A Simple Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation

Representation is an ordered list of city numbers known as an *order-based* GA.

- 1) London 3) Dunedin 5) Beijing 7) Tokyo
 2) Venice 4) Singapore 6) Phoenix 8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

43

Crossover

Crossover combines inversion and recombination:

		*		*		
Parent1	(3 5	7 2 1 6	4 8)			
Parent2	(2 5	7 6 8 1	3 4)			
Child	(5 8	7 2 1 6	3 4)			

This operator is called the *Order1* crossover.

44

Mutation

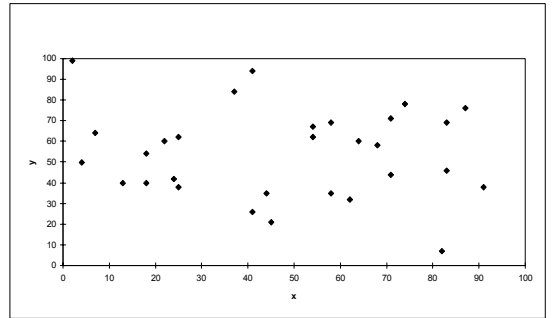
Mutation involves reordering of the list:

Before: (5 8 **7** 2 1 **6** 3 4)
After: (5 8 **6** 2 1 **7** 3 4)

The diagram shows a list of numbers: (5 8 7 2 1 6 3 4). The numbers 7 and 6 are highlighted with a grey background and an asterisk above them. A double-headed arrow points from 7 to 6, indicating a swap. In the 'After' state, the list is (5 8 6 2 1 7 3 4), with 6 and 7 highlighted and swapped.

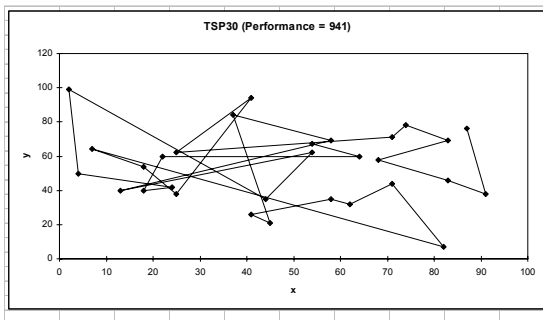
45

TSP Example: 30 Cities



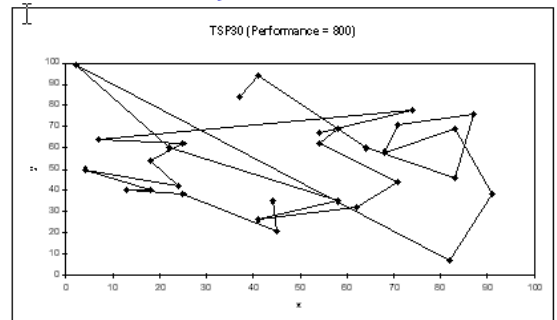
46

Solution i (Distance = 941)



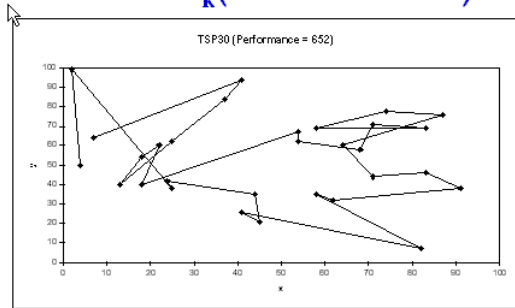
47

Solution j (Distance = 800)



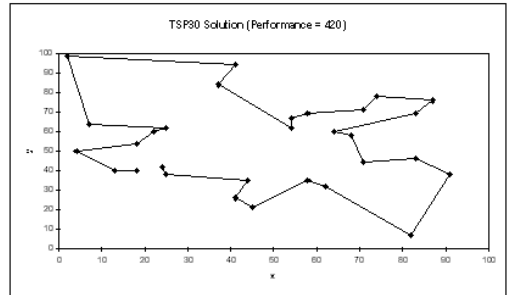
48

Solution k (Distance = 652)



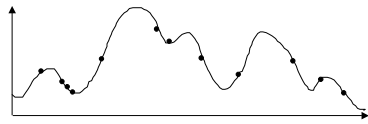
49

Best Solution (Distance = 420)

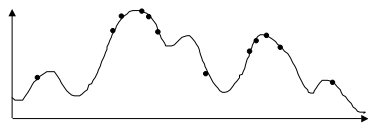


50

An Abstract Example



Distribution of Individuals in Generation 0



Distribution of Individuals in Generation N

51