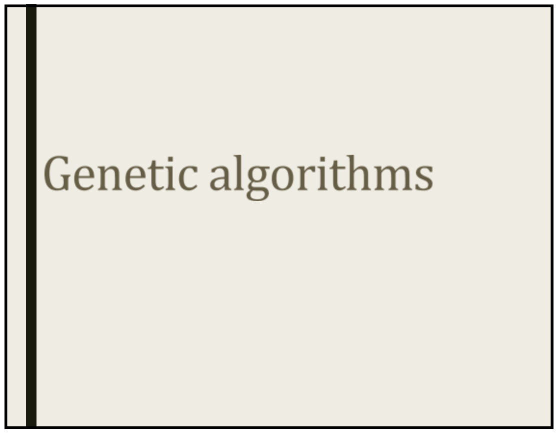
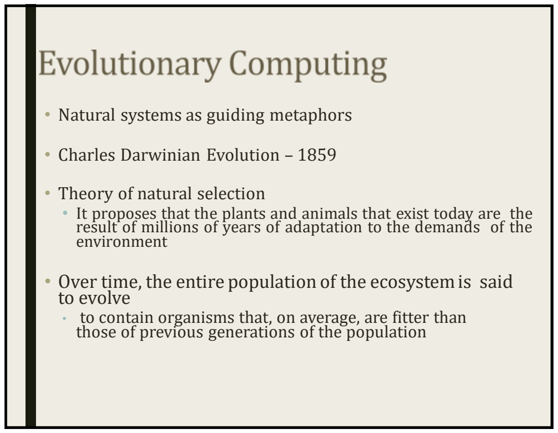


Genetic algorithms

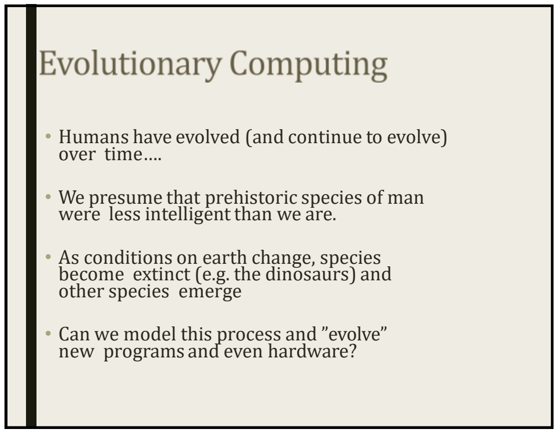
1



Evolutionary Computing

- Natural systems as guiding metaphors
- Charles Darwinian Evolution – 1859
- Theory of natural selection
  - It proposes that the plants and animals that exist today are the result of millions of years of adaptation to the demands of the environment
- Over time, the entire population of the ecosystem is said to evolve
  - to contain organisms that, on average, are fitter than those of previous generations of the population

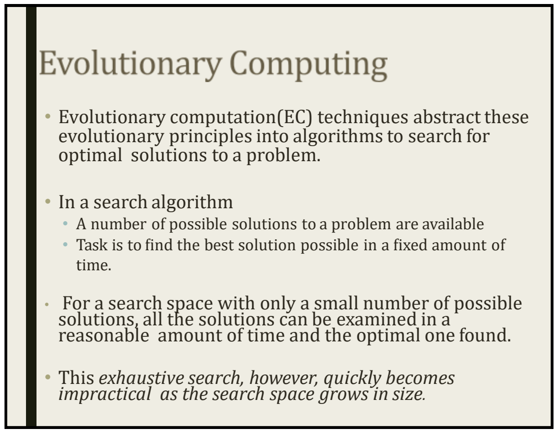
2



Evolutionary Computing

- Humans have evolved (and continue to evolve) over time….
- We presume that prehistoric species of man were less intelligent than we are.
- As conditions on earth change, species become extinct (e.g. the dinosaurs) and other species emerge
- Can we model this process and "evolve" new programs and even hardware?

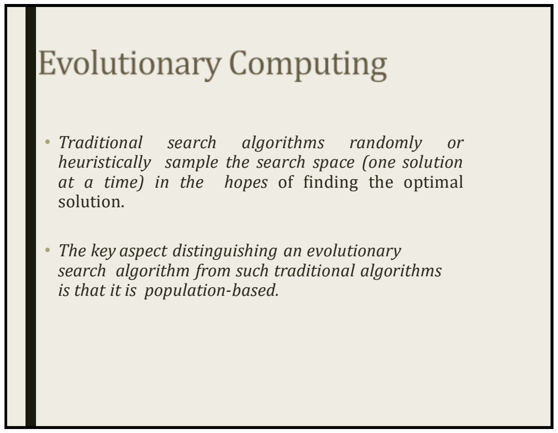
3



Evolutionary Computing

- Evolutionary computation(EC) techniques abstract these evolutionary principles into algorithms to search for optimal solutions to a problem.
- In a search algorithm
  - A number of possible solutions to a problem are available
  - Task is to find the best solution possible in a fixed amount of time.
- For a search space with only a small number of possible solutions, all the solutions can be examined in a reasonable amount of time and the optimal one found.
- This *exhaustive search, however, quickly becomes impractical as the search space grows in size.*

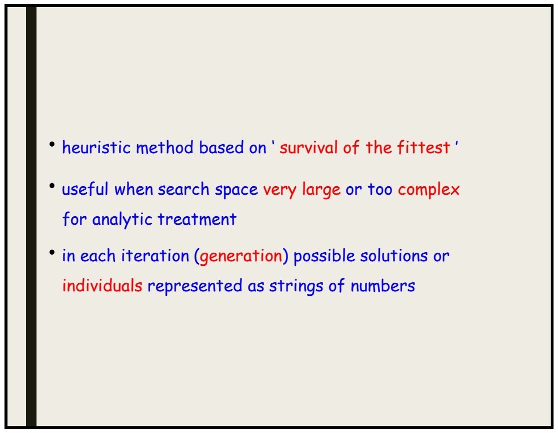
4



Evolutionary Computing

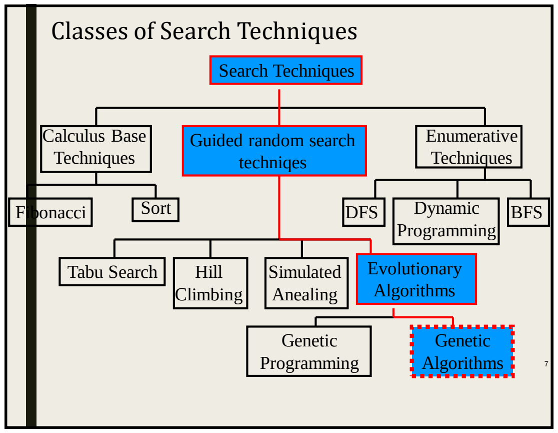
- *Traditional search algorithms randomly or heuristically sample the search space (one solution at a time) in the hopes* of finding the optimal solution.
- *The key aspect distinguishing an evolutionary search algorithm from such traditional algorithms is that it is population-based.*

5



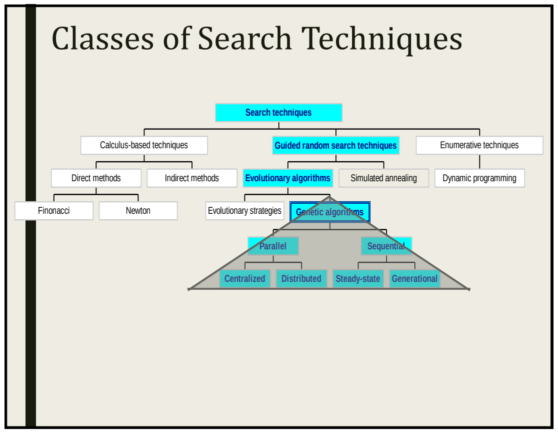
- heuristic method based on ' survival of the fittest '
- useful when search space very large or too complex for analytic treatment
- in each iteration (generation) possible solutions or individuals represented as strings of numbers

6



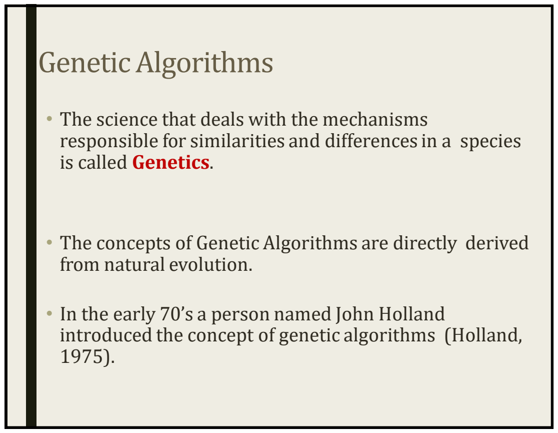
Classes of Search Techniques

7



Classes of Search Techniques

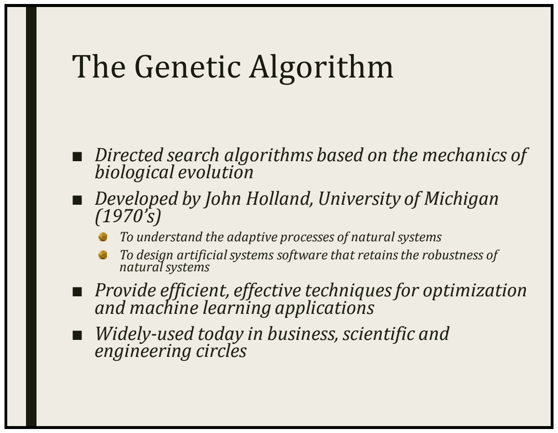
8



Genetic Algorithms

- The science that deals with the mechanisms responsible for similarities and differences in a species is called **Genetics**.
- The concepts of Genetic Algorithms are directly derived from natural evolution.
- In the early 70's a person named John Holland introduced the concept of genetic algorithms (Holland, 1975).

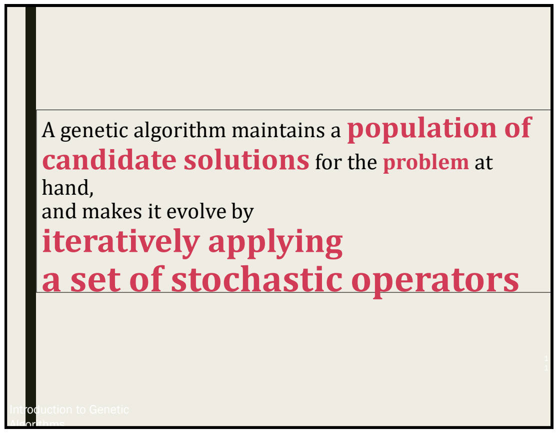
9



The Genetic Algorithm

- *Directed search algorithms based on the mechanics of biological evolution*
- *Developed by John Holland, University of Michigan (1970's)*
  - *To understand the adaptive processes of natural systems*
  - *To design artificial systems software that retains the robustness of natural systems*
- *Provide efficient, effective techniques for optimization and machine learning applications*
- *Widely-used today in business, scientific and engineering circles*

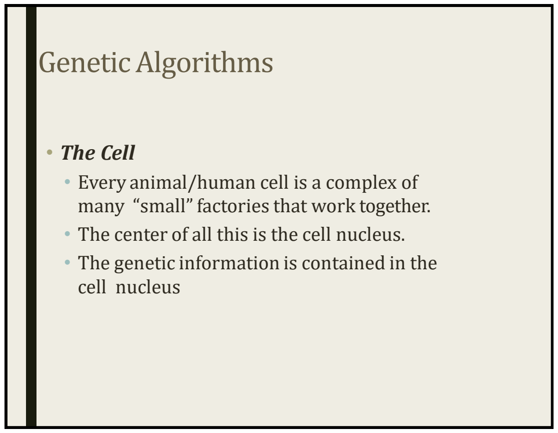
10



A genetic algorithm maintains a **population of candidate solutions** for the **problem** at hand,
and makes it evolve by
iteratively applying
a set of stochastic operators

Introduction to Genetic Algorithms

11



Genetic Algorithms

- ***The Cell***
  - Every animal/human cell is a complex of many "small" factories that work together.
  - The center of all this is the cell nucleus.
  - The genetic information is contained in the cell nucleus

12

Genetic Algorithms

- ***Chromosomes***
  - All the genetic information gets stored in the chromosomes.
  - The chromosomes are divided into several parts called genes.
    - Genes code the properties of species i.e., the characteristics of an individual.
    - The possibilities of the genes for one property are called allele and a gene can take different alleles.
    - For example, there is a gene for eye color, and all the different possible alleles are black, brown, blue and green

13

Genetic Algorithms

- Another biological based method.
- Based on the Darwin's theory. (Evolution of species).
- Based on: **survival of the most fittest individual**
- Key steps: *reproduction, survive*
- Try to simulate life.
- Individual = solution
- Environment = problem

14

The Main Stages of GA

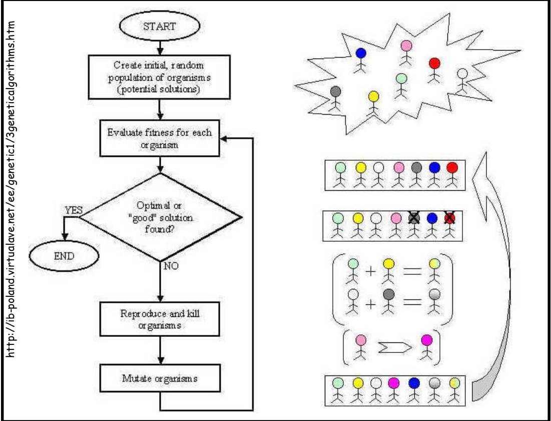
The main stages of GA are:

1. Creation of a **population** of individuals
2. **Evaluation** of individual's performance in the given environment
3. **Selection** of best individuals
4. Genetic **manipulation** on the selected individuals

- At stage 1, a population of individuals is created, and then the population is modified through a *cycle* of stages 2,..., 4

15

15


http://lb-poland.virtualave.net/ee/genetic1/3geneticalgorithms.htm
START
Create initial, random population of organisms (potential solutions)
Evaluate fitness for each organism
Optimal or "good" solution found?
YES
END
NO
Reproduce and kill organisms
Mutate organisms

16

Representation

- Like in real life, the process is not done on the solution (individual), but on its representation (**chromosome**).
- Therefore we don't use knowledge we have about the problem.
- Very often, based on strings. (but not always), like biological chromosomes.

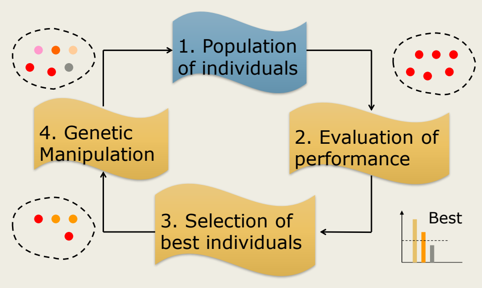
17

Breeding Ground

- The idea is that you represent the solution to your problem in a structured way
  - the simplest is a string
  - this is not necessarily the best representation, but it is simple to explain
- We then build a population of random solutions and let them "breed" using genetic operators
  - at each generation we apply "**survival of the fittest**" and hopefully better and better solutions evolve over time
  - the best solutions are more likely to survive and more likely to produce even better solutions

18

..Stages of GA


1. Population of individuals
2. Evaluation of performance
3. Selection of best individuals
4. Genetic Manipulation
Best

19

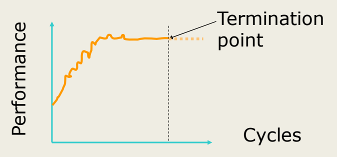
..Stages of GA

- Each cycle of GA produces a new population of individuals with better performance
- In the example of selecting red roses, this means that the proportion of red roses becomes greater and grey less (because the grey roses are not selected for the next seed)

20

Termination of GA

- The cycles run while the performance is improved
- Once the performance becomes stable (e.g., all roses became red), the GA terminates



21

Definitions for GA

- A population is often named *generation*
- A new population of individuals, created as a result of the genetic manipulation, is often called *offspring*
- The performance of individuals in the given environment are also called *fitness*
- For example, if our aim is select horses for racing, the performance (fitness) can be evaluated in terms of speed

22

..Definitions for GA

- Individuals can be coded into *bit-strings* which are also named *chromosomes*
- The bits are often named *genes*
- An example of string (chromosome) consisting of 5 bits (genes):

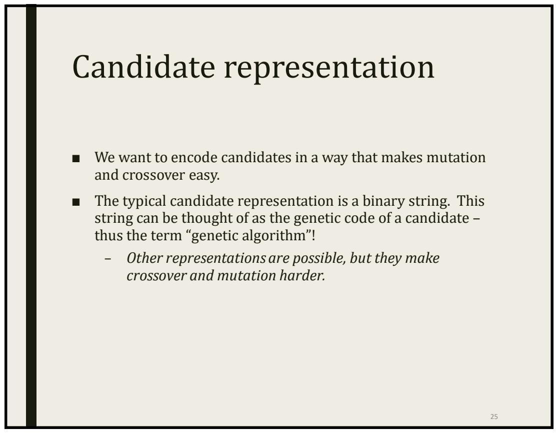
10010

23

Types of Genetic Manipulation

- Typically, genetic manipulation on selection of individuals consists of 2 operators: ***crossover* and *mutation***
- The crossover operator recombines bits of two selected strings at the *crossover point*
- The crossover points are assigned randomly among the bits of strings
- One or more crossover points can be given

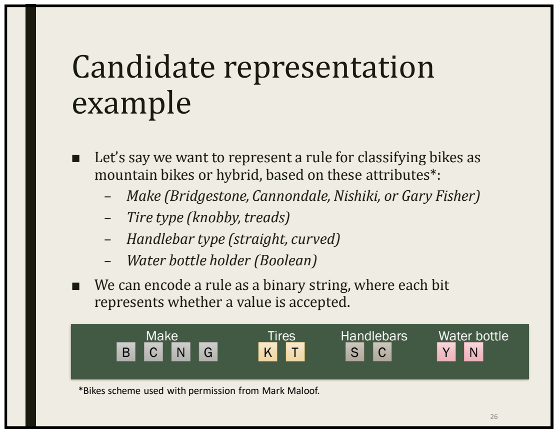
24



Candidate representation

- We want to encode candidates in a way that makes mutation and crossover easy.
- The typical candidate representation is a binary string. This string can be thought of as the genetic code of a candidate – thus the term "genetic algorithm"!
  - *Other representations are possible, but they make crossover and mutation harder.*

25



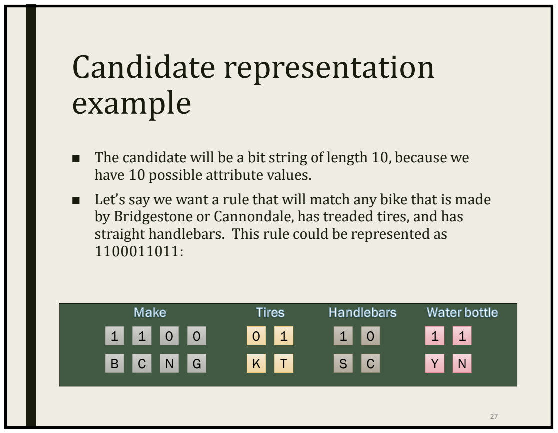
Candidate representation example

- Let's say we want to represent a rule for classifying bikes as mountain bikes or hybrid, based on these attributes*:
  - *Make (Bridgestone, Cannondale, Nishiki, or Gary Fisher)*
  - *Tire type (knobby, treads)*
  - *Handlebar type (straight, curved)*
  - *Water bottle holder (Boolean)*
- We can encode a rule as a binary string, where each bit represents whether a value is accepted.

Make				Tires		Handlebars		Water bottle	
B	C	N	G	K	T	S	C	Y	N

*Bikes scheme used with permission from Mark Maloof.

26

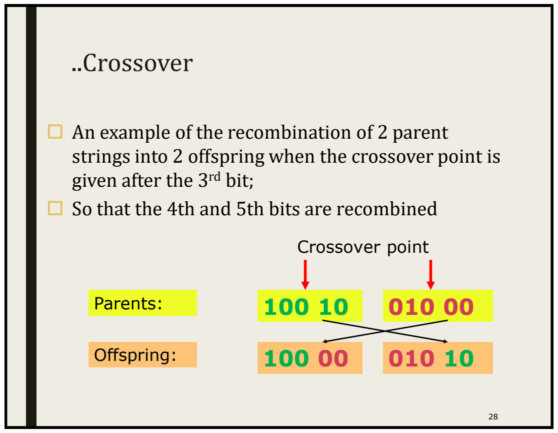


Candidate representation example

- The candidate will be a bit string of length 10, because we have 10 possible attribute values.
- Let's say we want a rule that will match any bike that is made by Bridgestone or Cannondale, has treaded tires, and has straight handlebars. This rule could be represented as 1100011011:

Make				Tires		Handlebars		Water bottle	
1	1	0	0	0	1	1	0	1	1
B	C	N	G	K	T	S	C	Y	N

27



..Crossover

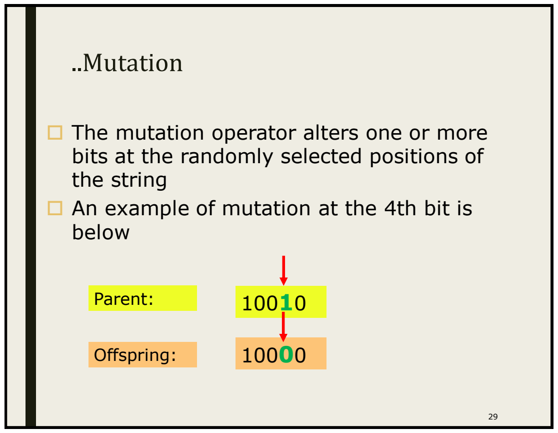
- An example of the recombination of 2 parent strings into 2 offspring when the crossover point is given after the 3rd bit;
- So that the 4th and 5th bits are recombined

Crossover point

Parents: 100 10 010 00

Offspring: 100 00 010 10

28



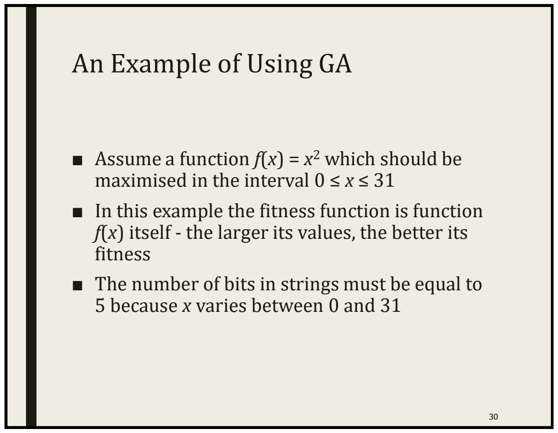
..Mutation

- The mutation operator alters one or more bits at the randomly selected positions of the string
- An example of mutation at the 4th bit is below

Parent: 10010

Offspring: 10000

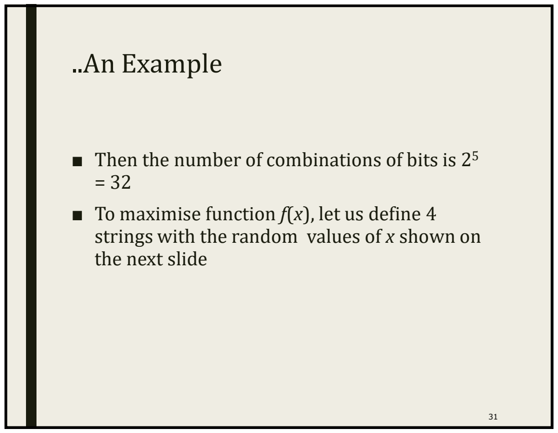
29



An Example of Using GA

- Assume a function $f(x) = x^2$ which should be maximised in the interval $0 \le x \le 31$
- In this example the fitness function is function $f(x)$ itself - the larger its values, the better its fitness
- The number of bits in strings must be equal to 5 because x varies between 0 and 31

30



..An Example

- Then the number of combinations of bits is 2^5 = 32
- To maximise function $f(x)$, let us define 4 strings with the random values of x shown on the next slide

31

31

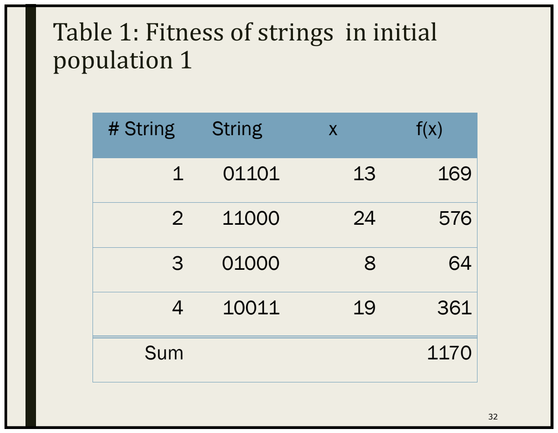
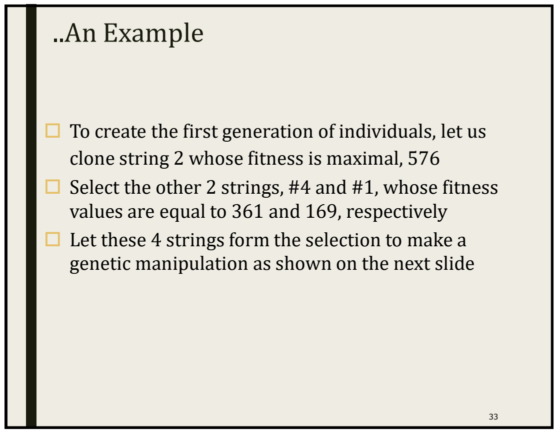


Table 1: Fitness of strings in initial population 1

# String	String	x	f(x)
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361
Sum			1170

32

32

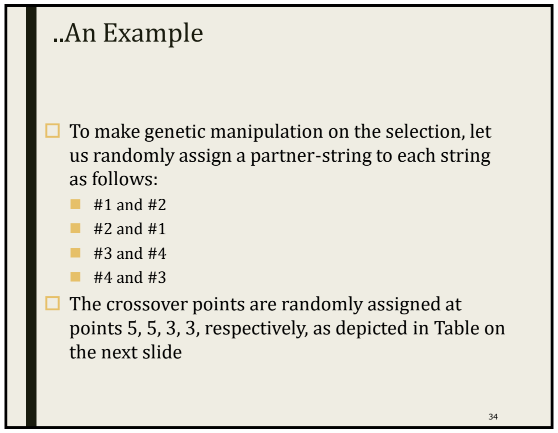


..An Example

- To create the first generation of individuals, let us clone string 2 whose fitness is maximal, 576
- Select the other 2 strings, #4 and #1, whose fitness values are equal to 361 and 169, respectively
- Let these 4 strings form the selection to make a genetic manipulation as shown on the next slide

33

33



..An Example

- To make genetic manipulation on the selection, let us randomly assign a partner-string to each string as follows:
  - #1 and #2
  - #2 and #1
  - #3 and #4
  - #4 and #3
- The crossover points are randomly assigned at points 5, 5, 3, 3, respectively, as depicted in Table on the next slide

34

34

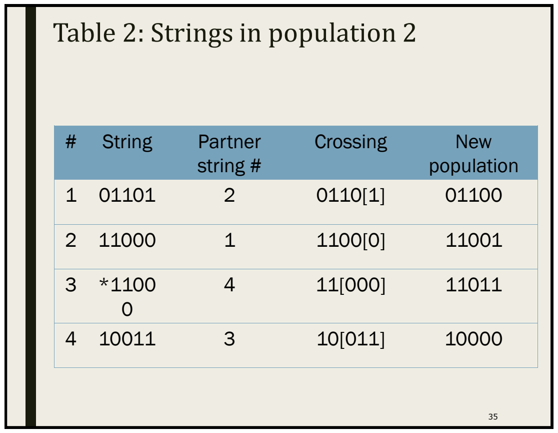
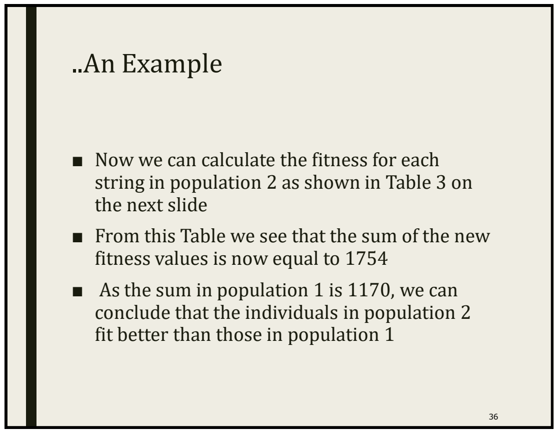


Table 2: Strings in population 2

#	String	Partner string #	Crossing	New population
1	01101	2	0110[1]	01100
2	11000	1	1100[0]	11001
3	*1100 0	4	11[000]	11011
4	10011	3	10[011]	10000

35

35



..An Example

- Now we can calculate the fitness for each string in population 2 as shown in Table 3 on the next slide
- From this Table we see that the sum of the new fitness values is now equal to 1754
- As the sum in population 1 is 1170, we can conclude that the individuals in population 2 fit better than those in population 1

36

36

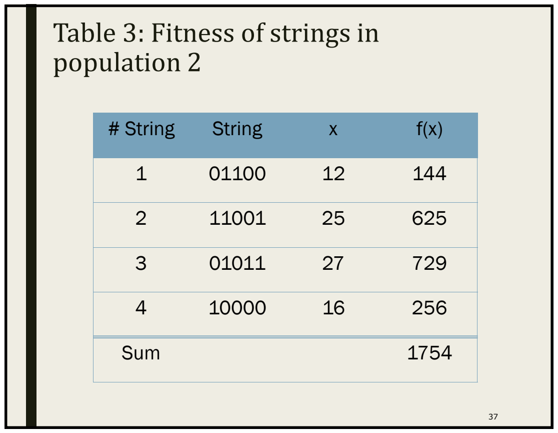
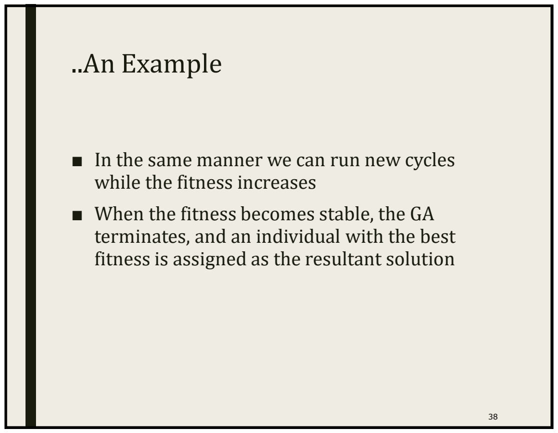


Table 3: Fitness of strings in population 2

# String	String	x	f(x)
1	01100	12	144
2	11001	25	625
3	01011	27	729
4	10000	16	256
Sum			1754

37

37

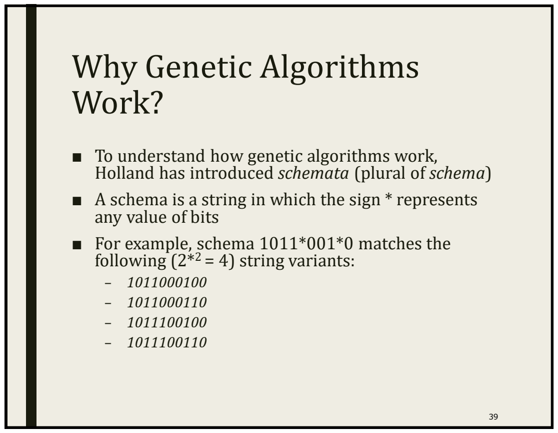


..An Example

- In the same manner we can run new cycles while the fitness increases
- When the fitness becomes stable, the GA terminates, and an individual with the best fitness is assigned as the resultant solution

38

38

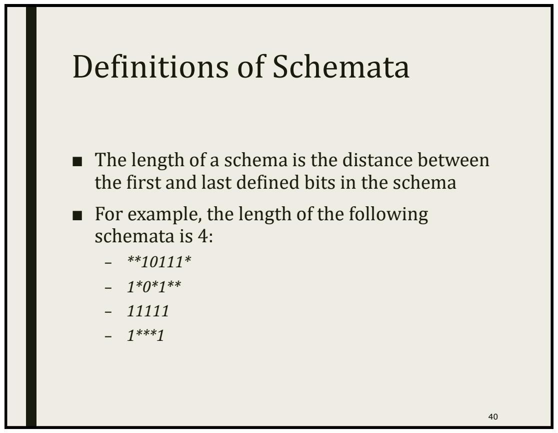


Why Genetic Algorithms Work?

- To understand how genetic algorithms work, Holland has introduced *schemata* (plural of *schema*)
- A schema is a string in which the sign * represents any value of bits
- For example, schema 1011*001*0 matches the following ($2^{*2} = 4$) string variants:
  - *1011000100*
  - *1011000110*
  - *1011100100*
  - *1011100110*

39

39

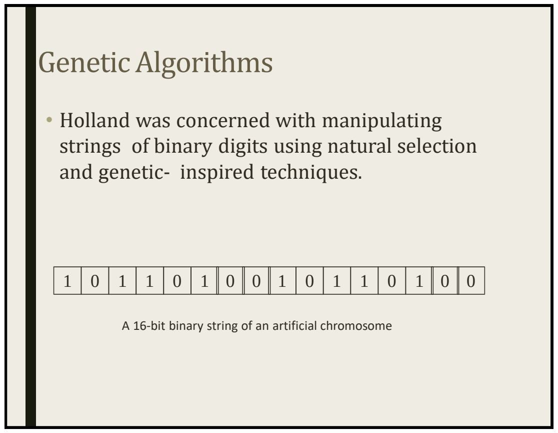


Definitions of Schemata

- The length of a schema is the distance between the first and last defined bits in the schema
- For example, the length of the following schemata is 4:
  - ***10111**
  - *1*0*1***
  - *11111*
  - *1***1*

40

40



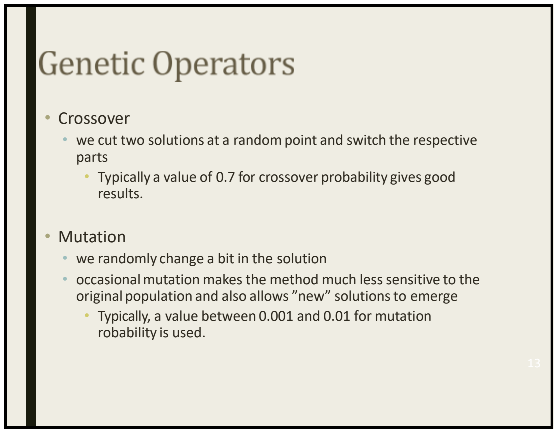
Genetic Algorithms

- Holland was concerned with manipulating strings of binary digits using natural selection and genetic- inspired techniques.

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

A 16-bit binary string of an artificial chromosome

41

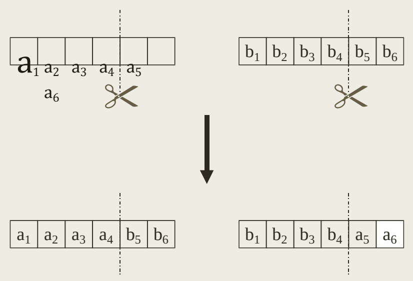


Genetic Operators

- Crossover
  - we cut two solutions at a random point and switch the respective parts
    - Typically a value of 0.7 for crossover probability gives good results.
- Mutation
  - we randomly change a bit in the solution
  - occasional mutation makes the method much less sensitive to the original population and also allows "new" solutions to emerge
    - Typically, a value between 0.001 and 0.01 for mutation robability is used.

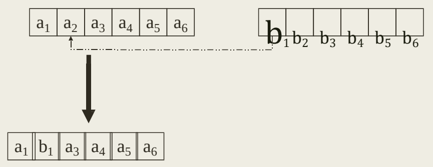
42

Diagram of Crossover



43

Diagram of Mutation



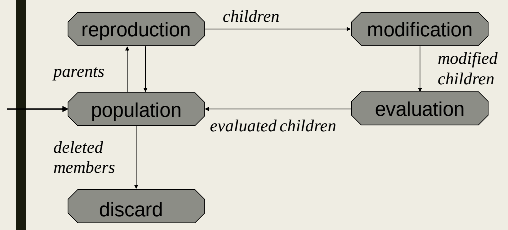
44

Stochastic operators

- ***Selection*** replicates the most successful solutions found in a population at a rate proportional to their relative quality
- ***Recombination*** decomposes two distinct solutions and then randomly mixes their parts to form novel solutions
- ***Mutation*** randomly perturbs a candidate solution

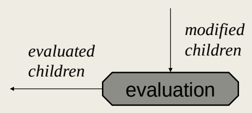
45

The GA Cycle of Reproduction



46

Evaluation



- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

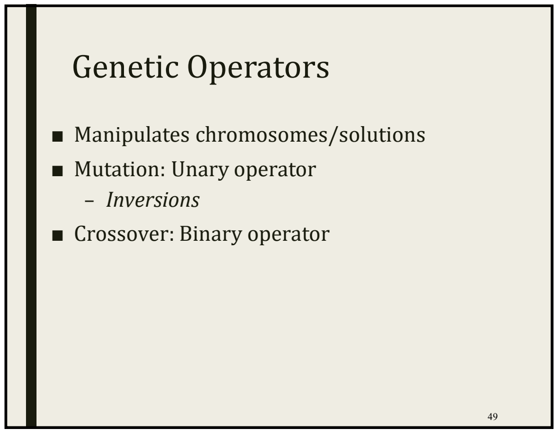
47

Evaluation of Individuals

- Adaptability – "fitness"
- Relates to the objective function value for a DOP
- Fitness is maximized
- Used in selection ("*Survival of the fittest*")
- Often *normalized*

$$f : S \to [0,1]$$

48

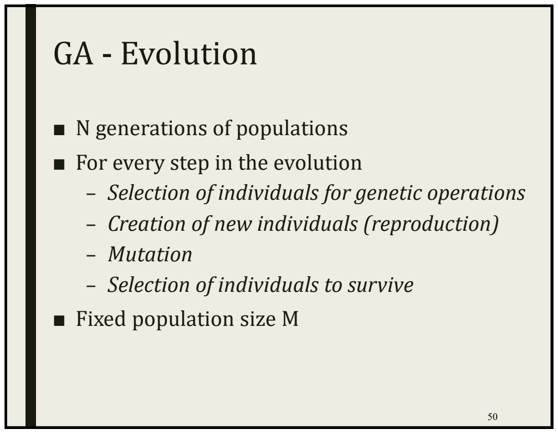


Genetic Operators

- Manipulates chromosomes/solutions
- Mutation: Unary operator
  - *Inversions*
- Crossover: Binary operator

49

49

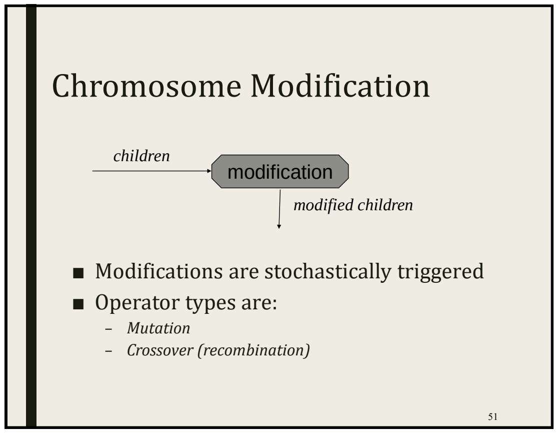


GA - Evolution

- N generations of populations
- For every step in the evolution
  - *Selection of individuals for genetic operations*
  - *Creation of new individuals (reproduction)*
  - *Mutation*
  - *Selection of individuals to survive*
- Fixed population size M

50

50



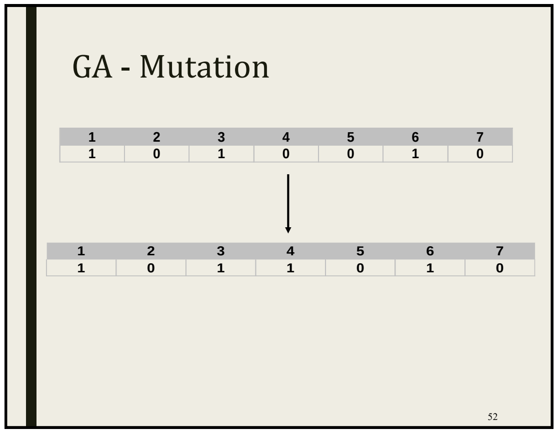
Chromosome Modification

children → modification → *modified children*

- Modifications are stochastically triggered
- Operator types are:
  - *Mutation*
  - *Crossover (recombination)*

51

51



GA - Mutation

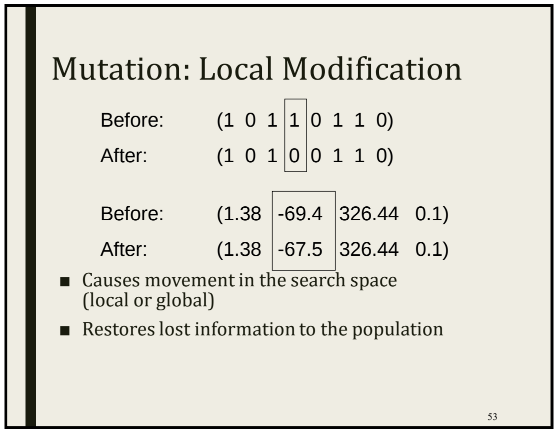
1	2	3	4	5	6	7
1	0	1	0	0	1	0

↓

1	2	3	4	5	6	7
1	0	1	1	0	1	0

52

52



Mutation: Local Modification

Before: (1 0 1 $\boxed{1}$ 0 1 1 0)

After: (1 0 1 $\boxed{0}$ 0 1 1 0)

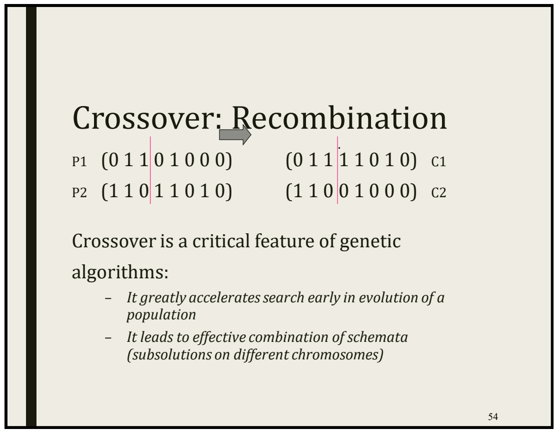
Before: (1.38 $\boxed{-69.4}$ 326.44 0.1)

After: (1.38 $\boxed{-67.5}$ 326.44 0.1)

- Causes movement in the search space (local or global)
- Restores lost information to the population

53

53



Crossover: Recombination

P1 (0 1 1 | 0 1 0 0 0) ⇒ (0 1 1 | 1 1 0 1 0) C1

P2 (1 1 0 | 1 1 0 1 0) ⇒ (1 1 0 | 0 1 0 0 0) C2

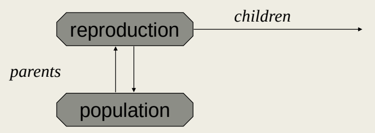
Crossover is a critical feature of genetic algorithms:

- *It greatly accelerates search early in evolution of a population*
- *It leads to effective combination of schemata (subsolutions on different chromosomes)*

54

54

Reproduction

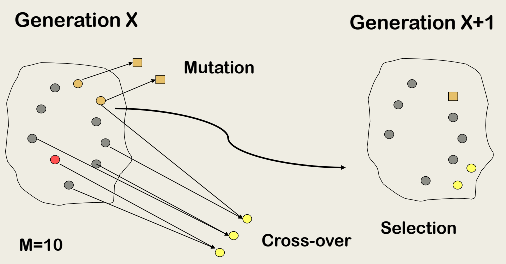


Parents are selected at random with selection chances biased in relation to chromosome evaluations

55

55

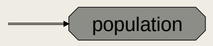
GA - Evolution



56

56

Population



Chromosomes could be:

– *Bit strings*	*(0101 ... 1100)*
– *Real numbers*	*(43.2 -33.1 ... 0.0 89.2)*
– *Permutations of element*	*(E11 E3 E7 ... E1 E15)*
– *Lists of rules*	*(R1 R2 R3 ... R22 R23)*
– *Program elements*	*(genetic programming)*
– *... any data structure ...*	

57

57

Classical GA: Binary chromosomes

1	2	3	4	5	6	7
1	0	1	0	0	1	0

1	2	3	4	5	6	7
0	1	1	1	0	0	1

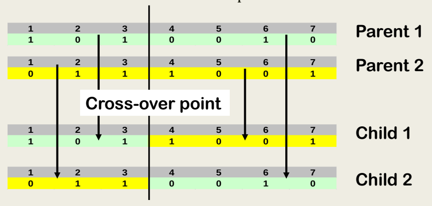
- Functional optimization
  - *Chromosome corresponds to a binary encoding of a real number - min/max of an arbitrary function*
- COP, TSP as an example
  - *Binary encoding of a solution*
  - *Often better with a more direct representation (e.g. sequence representation)*

58

58

GA - Classical Crossover (1-point)

- One parent is selected based on *fitness*
- The other parent is selected randomly
- Random choice of cross-over point



Parent 1

1	2	3	4	5	6	7
1	0	1	0	0	1	0

Parent 2

1	2	3	4	5	6	7
0	1	1	1	0	0	1

Child 1

1	2	3	4	5	6	7
1	0	1	1	0	0	1

Child 2

1	2	3	4	5	6	7
0	1	1	0	0	1	0

59

59

GA – Classical Crossover

- Arbitrary (or worst) individual in the population is changed with one of the two offspring (e.g. the best)
- Reproduce as long as you want
- Can be regarded as a sequence of almost equal populations
- Alternatively:
  - *One parent selected according to fitness*
  - *Crossover until (at least) M offspring are created*
  - *The new population consists of the offspring*
- Lots of other possibilities ...
- Basic GA with classical crossover and mutation often works well

60

60

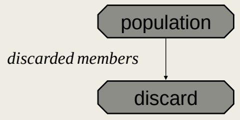
GA – Standard Reproduction Plan

- Fixed population size
- Standard cross-over
  - *One parent selected according to fitness*
  - *The other selected randomly*
  - *Random cross-over point*
  - *A random individual is exchanged with one of the offspring*
- Mutation
  - *A certain probability that an individual mutate*
  - *Random choice of which gene to mutate*
  - *Standard: mutation of offspring*

61

61

Deletion

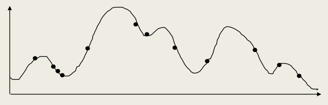


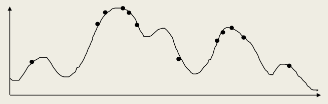
- *Generational* GA:
  entire populations replaced each iteration
- *Steady-state* GA:
  a few members replaced each generation

62

62

An Abstract Example


Distribution of Individuals in Generation 0


Distribution of Individuals in Generation N

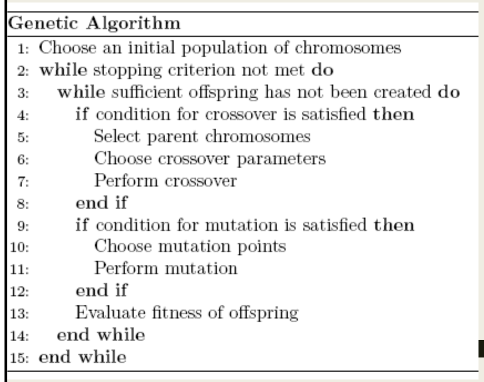
63

63

Simple Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

64



Genetic Algorithm

```
Choose an initial population of chromosomes
while stopping criterion not met do
    while sufficient offspring has not been created do
        if condition for crossover is satisfied then
            Select parent chromosomes
            Choose crossover parameters
            Perform crossover
        end if
        if condition for mutation is satisfied then
            Choose mutation points
            Perform mutation
        end if
        Evaluate fitness of offspring
    end while
end while
```

65

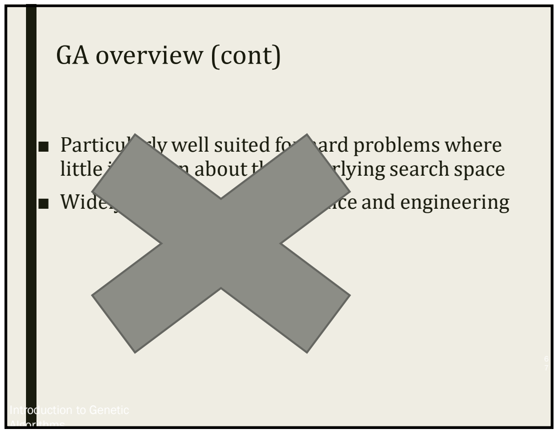
65

Genetic Algorithms (GA) OVERVIEW

- A class ilistic o n algorithms
- Inspired on process
- Uses concept ection" and "Genetic Inheritance
- Origin nd (1975)

Introduction to Genetic Algorithms

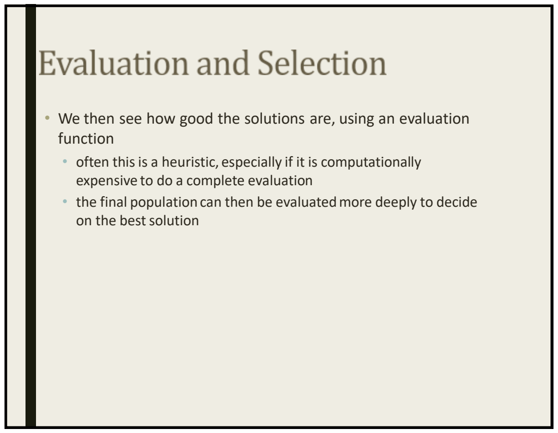
66



GA overview (cont)

- Particu ly well suited fo ard problems where little i n about th rlying search space
- Wide ce and engineering

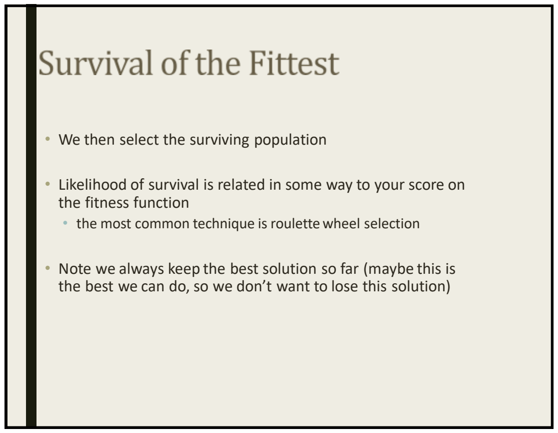
67



Evaluation and Selection

- We then see how good the solutions are, using an evaluation function
  - often this is a heuristic, especially if it is computationally expensive to do a complete evaluation
  - the final population can then be evaluated more deeply to decide on the best solution

68

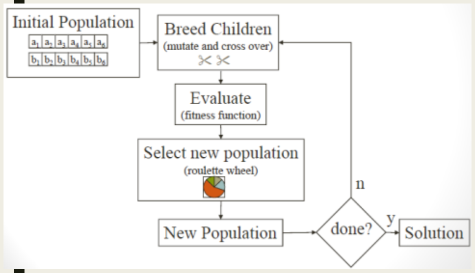


Survival of the Fittest

- We then select the surviving population
- Likelihood of survival is related in some way to your score on the fitness function
  - the most common technique is roulette wheel selection
- Note we always keep the best solution so far (maybe this is the best we can do, so we don't want to lose this solution)

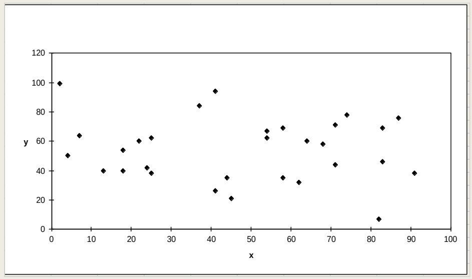
69

Schematic of GA



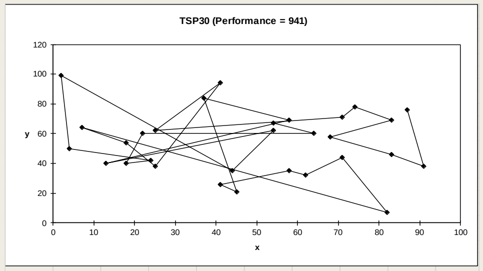
70

TSP Example: 30 Cities



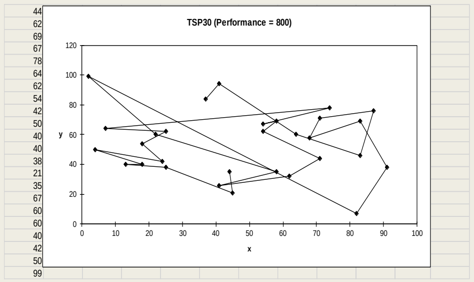
71

Solution $_i$ (Distance = 941)



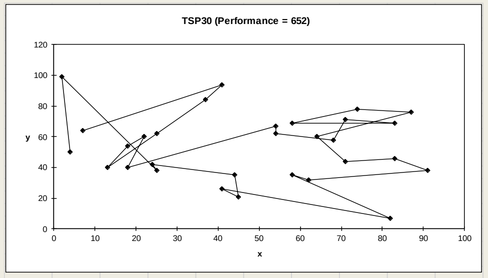
72

Solution $_j$ (Distance = 800)



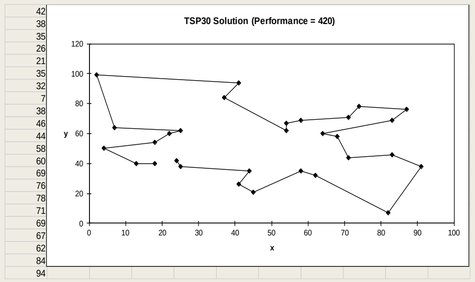
73

Solution $_k$ (Distance = 652)



74

Best Solution (Distance = 420)



75

Problem Definition

The **Traveling Salesman Problem** involves finding the shortest possible route that visits each city exactly once and returns to the origin city. Let's consider a small example with 5 cities.

Cities and Distances

We'll define 5 cities (A, B, C, D, E) and their distances in a matrix form:

	A	B	C	D	E
A	0	2	9	10	7
B	2	0	6	4	3
C	9	6	0	8	5
D	10	4	8	0	6
E	7	3	5	6	0

76

Step 1: Initialization

First, we generate an initial population of chromosomes. Each chromosome represents a possible tour. Let's assume a population size of 4.

Chromosome	Tour	Distance
C1	A-B-C-D-E-A	27
C2	A-C-B-E-D-A	38
C3	A-D-E-B-C-A	31
C4	A-E-D-C-B-A	36

Step 2: Selection

We use roulette wheel selection to choose parents for reproduction. The probability of selecting a chromosome is inversely proportional to its tour distance.

Chromosome	Distance	Fitness (1/Distance)	Probability
C1	27	0.037	0.364
C2	38	0.026	0.255
C3	31	0.032	0.291
C4	36	0.028	0.290

77

Step 3: Crossover

We perform crossover between selected parents to create offspring. Assume C1 and C3 are selected as parents, and C2 and C4 are selected as another pair of parents. We'll use partially matched crossover (PMX).

Crossover between C1 (A-B-C-D-E) and C3 (A-D-E-B-C):

Assume the crossover points are between the second and fourth cities:

- Parent 1: A-B-C-D-E
- Parent 2: A-D-E-B-C

Offspring:

- Offspring 1: A-B-E-D-C
- Offspring 2: A-D-C-B-E

Crossover between C2 (A-C-B-E-D) and C4 (A-E-D-C-B):

Assume the crossover points are between the first and third cities:

- Parent 1: A-C-B-E-D
- Parent 2: A-E-D-C-B

Offspring:

- Offspring 3: A-C-D-E-B
- Offspring 4: A-E-B-C-D

78

Step 4: Mutation
We apply mutation to introduce diversity. Assume a mutation rate of 0.1 (10%). We'll swap two cities in each chromosome.
Offspring 1 (A-B-E-D-C) with mutation between cities B and D:
•A-D-E-B-C
Offspring 2 (A-D-C-B-E) with mutation between cities C and B:
•A-D-B-C-E
Offspring 3 (A-C-D-E-B) with mutation between cities C and E:
•A-E-D-C-B
Offspring 4 (A-E-B-C-D) with mutation between cities B and D:
•A-E-D-C-B

Step 5: Replacement
We replace the old population with the new offspring.

Chromosome	Tour	Distance
O1	A-D-E-B-C-A	35
O2	A-D-B-C-E-A	33
O3	A-E-D-C-B-A	38
O4	A-E-D-C-B-A	38

79

Step 6: Termination
We repeat the selection, crossover, mutation, and replacement steps for a predefined number of generations or until a stopping criterion is met (e.g., a solution with a satisfactory fitness is found).

Generation 0
C1: A-B-C-D-E-A (27)
C2: A-C-B-E-D-A (38)
C3: A-D-E-B-C-A (31)
C4: A-E-D-C-B-A (36)

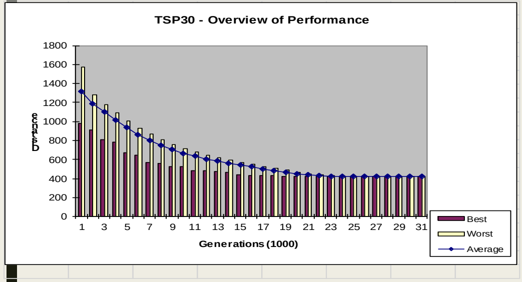
Selected parents: C1, C3, C2, C4
Crossover:
Parent 1: A-B-C-D-E, Parent 2: A-D-E-B-C $\longrightarrow$ Offspring 1: A-B-E-D-C, Offspring 2: A-D-C-B-E
Parent 1: A-C-B-E-D, Parent 2: A-E-D-C-B $\longrightarrow$ Offspring 3: A-C-D-E-B, Offspring 4: A-E-B-C-D

Mutation:
Offspring 1: A-B-E-D-C $\longrightarrow$ A-D-E-B-C
Offspring 2: A-D-C-B-E $\longrightarrow$ A-D-B-C-E
Offspring 3: A-C-D-E-B $\longrightarrow$ A-E-D-C-B
Offspring 4: A-E-B-C-D $\longrightarrow$ A-E-D-C-B

Generation 1
O1: A-D-E-B-C-A (35)
O2: A-D-B-C-E-A (33)
O3: A-E-D-C-B-A (38)
O4: A-E-D-C-B-A (38)

80

Overview of Performance



81

Example (Negnevitsky, 2002)

- To find the maximum value of the function ($15x$-x^2), where parameter x varies between 0 and 15. Assumption: x takes only integer values.

Integer	Binary code
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

82

Example (Contd.)

- Suppose that the size of the chromosome population N is 6, the crossover probability p_c equals 0.7 and the mutation probability p_m equals 0.001, the fitness function in our example is defined by

  $f(x) = (15x$-$x^2)$

83

Example (Contd.)

- The GA creates an initial population of chromosomes by filling six 4-bit strings with randomly generated ones and zeros.

Chromosome label	Chromosome string	Decoded integer	Chromosome fitness	Fitness ratio %
X1	1100	12	36	16.5
X2	0100	4	44	20.2
X3	0001	1	14	6.4
X4	1110	14	14	6.4
X5	0111	7	56	25.7
X6	1001	9	54	24.8

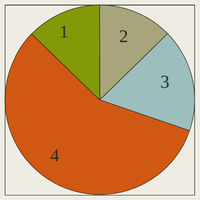
84

Example (Contd.)

- Last column in the previous table shows the ratio of the individual chromosome fitness to the populations total fitness.
- According to the example, chromosomes X5 and X6 have a pretty good chance of being selected for mating, while X3 and X4 have a poor chance.
- Roulette selection wheel is the most commonly used chromosome selection technique.

85

Roulette selection wheel



•Each member of the population is given a slice of the roulette wheel
•The better the fitness, the bigger the slice
•We then spin the wheel - if your number comes up you survive to the next generation!

86

Example (Contd.)

- To select a chromosome for mating, a random number (between 0 and 100) is generated.
- The roulette wheel is spun and when the arrow comes to rest on one of the segments, the corresponding chromosome is selected.
- In the current case, the wheel is spun six times.
  - The first two spins might select, X6 and X2;
  - Second two spins might select X1 and X5;
  - The last two spins might select X2 and X5.

87

Example (Contd.)

- Genetic algorithms assure the continous improvement of the average fitness of the population, and after many generations (typically several hundreds), the population evolves to a near optimal solution.
- In the current case, 0111 and 1000.

88

GA: Hybridization and Parallelization

- GAs strengths and weaknesses:
  - *Domain independence*
- Hybridization
  - *Seed good individuals in the initial population*
  - *Combine with other Metaheuristics to improve some solutions*
- Parallelization
  - *Fitness-evaluation*
  - *Sub-populations*
  - *The Island Model*

89

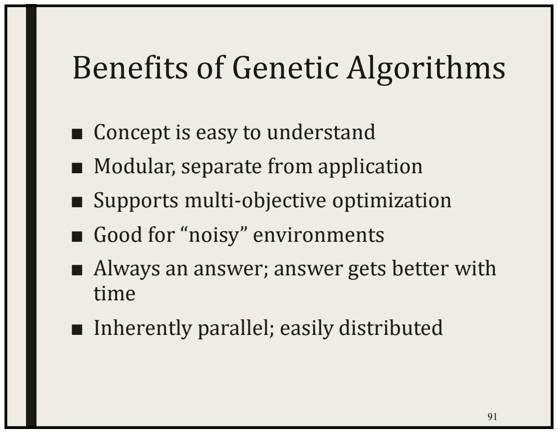
89

Issues for GA Practitioners

- Basic implementation issues:
  - *Representation*
  - *Population size, mutation rate, …*
  - *Selection, deletion policies*
  - *Crossover, mutation operators*
- Termination Criteria
- Performance, scalability
- Solution is only as good as the evaluation function (often hardest part)

90

90

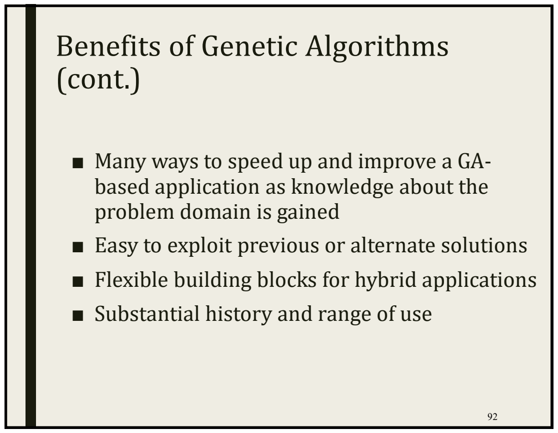


Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for "noisy" environments
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed

91

91

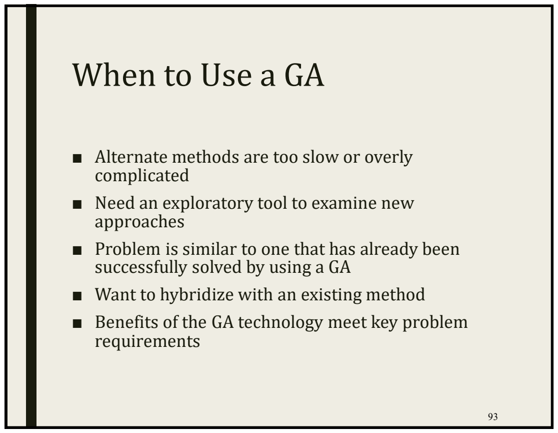


Benefits of Genetic Algorithms (cont.)

- Many ways to speed up and improve a GA-based application as knowledge about the problem domain is gained
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications
- Substantial history and range of use

92

92

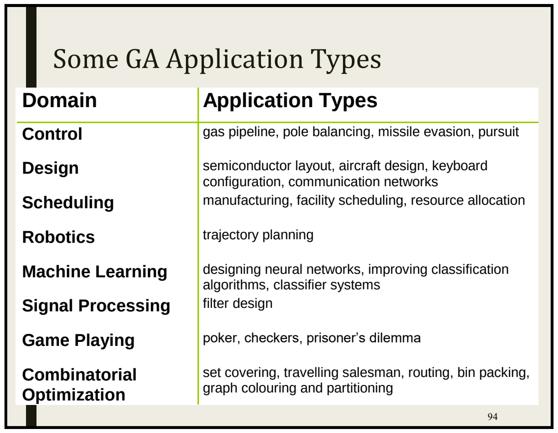


When to Use a GA

- Alternate methods are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing method
- Benefits of the GA technology meet key problem requirements

93

93

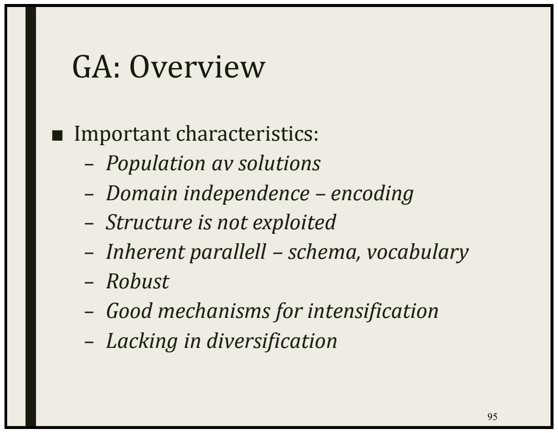


Some GA Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

94

94

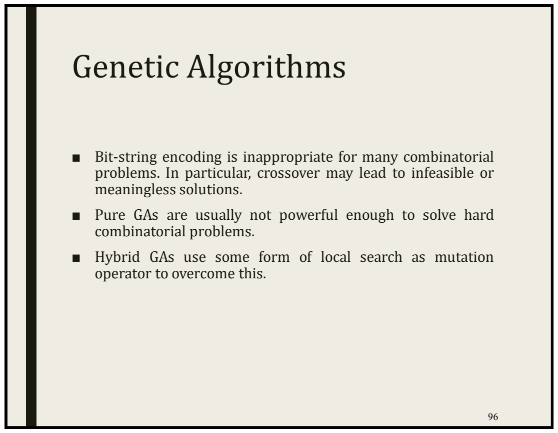


GA: Overview

- Important characteristics:
  - *Population av solutions*
  - *Domain independence – encoding*
  - *Structure is not exploited*
  - *Inherent parallell – schema, vocabulary*
  - *Robust*
  - *Good mechanisms for intensification*
  - *Lacking in diversification*

95

95

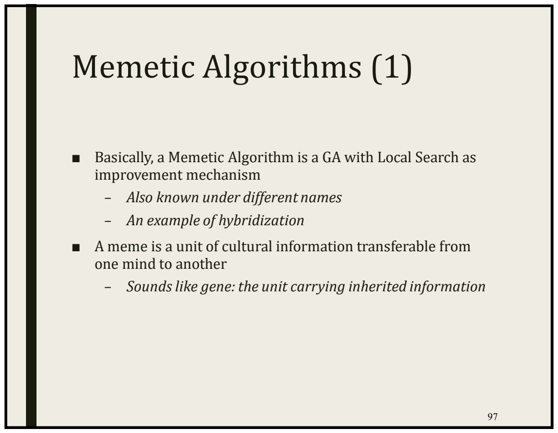


Genetic Algorithms

- Bit-string encoding is inappropriate for many combinatorial problems. In particular, crossover may lead to infeasible or meaningless solutions.
- Pure GAs are usually not powerful enough to solve hard combinatorial problems.
- Hybrid GAs use some form of local search as mutation operator to overcome this.

96

96

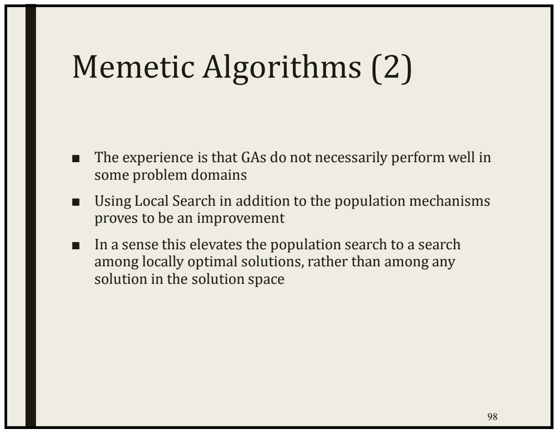


Memetic Algorithms (1)

- Basically, a Memetic Algorithm is a GA with Local Search as improvement mechanism
  - *Also known under different names*
  - *An example of hybridization*
- A meme is a unit of cultural information transferable from one mind to another
  - *Sounds like gene: the unit carrying inherited information*

97

97

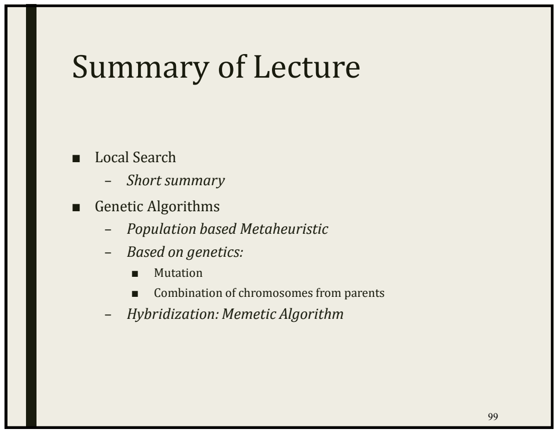


Memetic Algorithms (2)

- The experience is that GAs do not necessarily perform well in some problem domains
- Using Local Search in addition to the population mechanisms proves to be an improvement
- In a sense this elevates the population search to a search among locally optimal solutions, rather than among any solution in the solution space

98

98

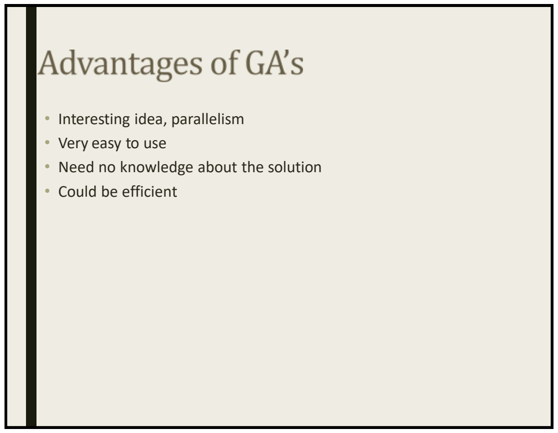


Summary of Lecture

- Local Search
  - *Short summary*
- Genetic Algorithms
  - *Population based Metaheuristic*
  - *Based on genetics:*
    - Mutation
    - Combination of chromosomes from parents
  - *Hybridization: Memetic Algorithm*

99

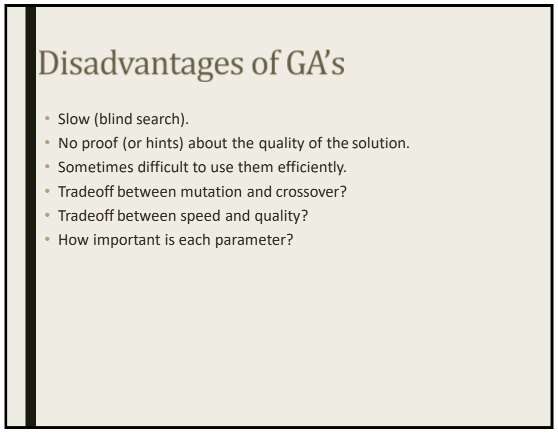
99



Advantages of GA's

- Interesting idea, parallelism
- Very easy to use
- Need no knowledge about the solution
- Could be efficient

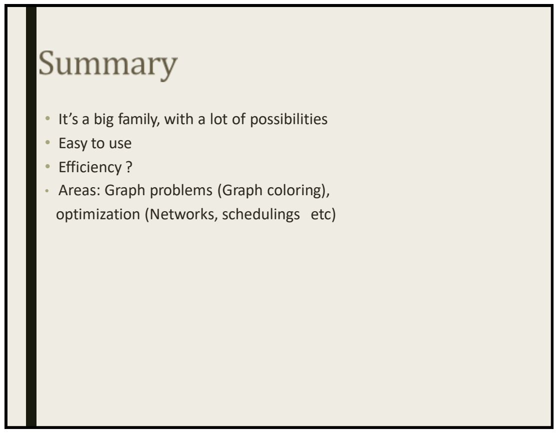
100



Disadvantages of GA's

- Slow (blind search).
- No proof (or hints) about the quality of the solution.
- Sometimes difficult to use them efficiently.
- Tradeoff between mutation and crossover?
- Tradeoff between speed and quality?
- How important is each parameter?

101



Summary

- It's a big family, with a lot of possibilities
- Easy to use
- Efficiency ?
- Areas: Graph problems (Graph coloring), optimization (Networks, schedulings etc)

102