Soft Computing Paradigm

Dr Sujit Das
Assistant Professor
Dept. of Computer Science and Engineering
NIT Warangal

What is Soft Computing?

- The idea behind soft computing is to model cognitive behavior of human mind.
- Soft computing is foundation of conceptual intelligence in machines.
- Unlike hard computing , Soft computing is tolerant of imprecision, uncertainty, partial truth, and approximation.

Hard Vs Soft Computing Paradigms

- Hard computing
  - Based on the concept of precise modeling and analyzing to yield accurate results.
  - Works well for simple problems, but is bound by the NP-Complete set.
- Soft computing
  - Aims to surmount NP-complete problems.
  - Uses inexact methods to give useful but inexact answers to intractable problems.
  - Represents a significant paradigm shift in the aims of computing - a shift which reflects the human mind.
  - Tolerant to imprecision, uncertainty, partial truth, and approximation.
  - Well suited for real world problems where ideal models are not available.

Difference b /w Soft and Hard Computing

Hard Computing	Soft Computing
Conventional computing requires a precisely stated analytical model.	Soft computing is tolerant of imprecision.
Often requires a lot of computation time.	Can solve some real world problems in reasonably less time.
Not suited for real world problems for which ideal model is not present.	Suitable for real world problems.
It requires full truth	Can work with partial truth
It is precise and accurate	Imprecise.
High cost for solution	Low cost for solution

Unique Features of Soft Computing

- Soft Computing is an approach for constructing systems which are
  - computationally intelligent,
  - possess human like expertise in particular domain,
  - can adapt to the changing environment and can learn to do better
  - can explain their decisions

Components of Soft Computing

- Components of soft computing include:
  - Fuzzy Logic (FL)
  - Evolutionary Computation (EC) - based on the origin of the species
    - Genetic Algorithm
    - Swarm Intelligence
    - Ant Colony Optimizations
  - Neural Network (NN)
  - Machine Learning (ML)

Evolutionary Computation

Genetic and Swarm Computing

Evolutionary Computation –EC

- General term for several computational techniques inspired by biological evolution
- Mostly involve meta-heuristic optimization algorithms such as:
  - Evolutionary algorithms
    - comprising genetic algorithms, evolutionary programming, etc)
  - Swarm intelligence
    - comprising ant colony optimization and particle swarm optimization)

Advantages of EC

- Conceptual Simplicity
- Broad Applicability
- Hybridization with Other Methods
- Parallelism
- Robust to Dynamic Changes
- Solves Problems that have no Solutions

GAs: A quick Overview

- Developed by John Holland, his colleagues, and his students, K. DeJong, D. Goldberg, University of Michigan (1970's), USA
  - to understand the **adaptive processes** of **natural systems**
  - to design artificial systems software that retains the **robustness of natural systems**

Why GAs

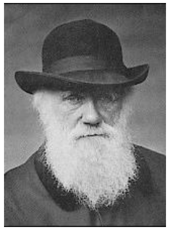
- Most of the real life problems are very complex and can not be solved in polynomial time using a deterministic algorithm.

- Sometimes attainment to the best is less important for complex problem. Rather **near optimal solutions** that can be generated quickly are more desirable than optimal solutions which require huge amount of time.
  - Ex: Traveling Salesman Problem (TSP).

  GAs strive to improve as close as possible to optimality.

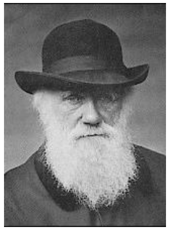
- When the problem can be modeled as an **optimization** one.

Why GAs (contd.)



- Not limited by restrictive assumptions about the search space
  - Continuity
  - Existence of derivative
  - Uni-modality and local minima
- Robust search in complex space
  - Balance between **efficiency** and **efficacy** (flexibility of biological system necessary for survival in many different situations)

Why GAs (contd.)



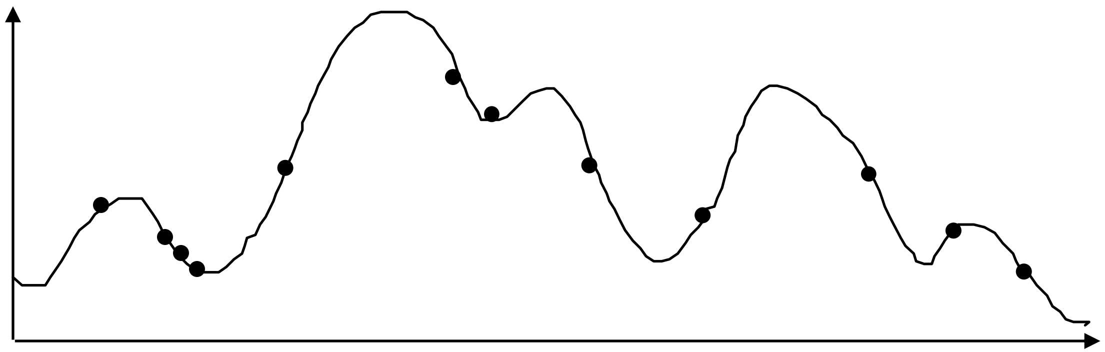
- Efficiently exploit historical information to speculate on new search points with expected improved performance : nature is full of precedents

- Adaptive : Features of self-repair, self-guidance and reproduction are the rules of the biological systems

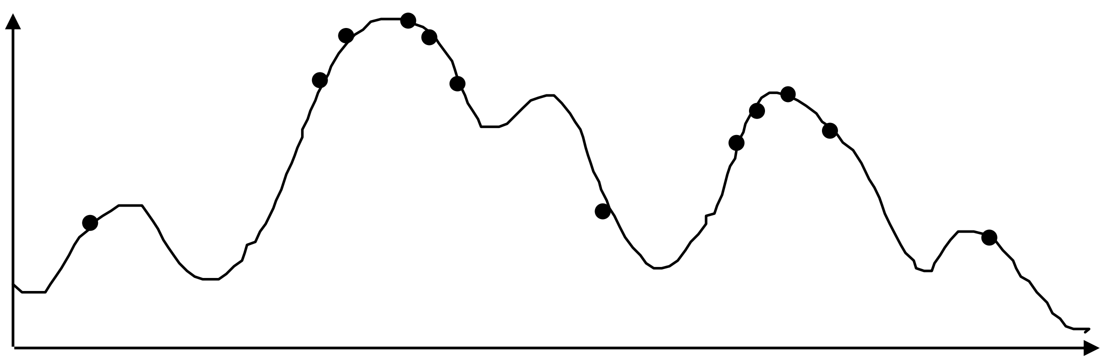
- Costly redesign can be reduced

GAs work from a rich database of points simultaneously, climbing many peaks in parallel; thus probability of finding a false peak (local optima) is reduced.

GAs

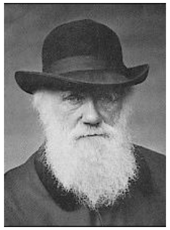


Distribution of Individuals in Generation 0



Distribution of Individuals in Generation N

Why GAs (contd.)



- Always an answer/solution; answer gets better with time
  empirically **proved to be convergent**
- Inherently parallel; easily distributed
- Supports multi-objective optimization
- Modular
- Finally, the power of GAs comes from **simple concepts**; easy to understand

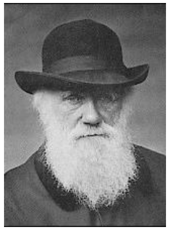
Genetic Algorithms

- Genetic algorithms are inspired by Darwin's theory of natural evolution.
- In the natural world, organisms that are poorly suited for an environment die off, while those well-suited, prosper.
- Genetic algorithms search the space of individuals for good candidates.
- The chance of an individual's being selected is proportional to the amount by which its fitness is greater or less than its competitors' fitness.

Contd..

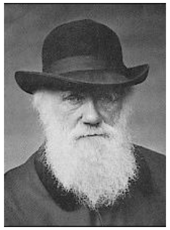
- Algorithm begins with a **set of initial solutions** (represented by set of **chromosomes**) called **population**.
- A **chromosome** is a string of elements called ***genes***.
- Solutions from one population are taken and are used to form a new population by generating offsprings.
- New population is formed using old population and offspring based on their fitness value.
- Promising candidates are kept and allowed to reproduce
- This is motivated by a hope, that the new population will be better than the old one.
- Genetic algorithms are broadly applicable and have the advantage that they require little knowledge encoded in the system.

Features of GAs

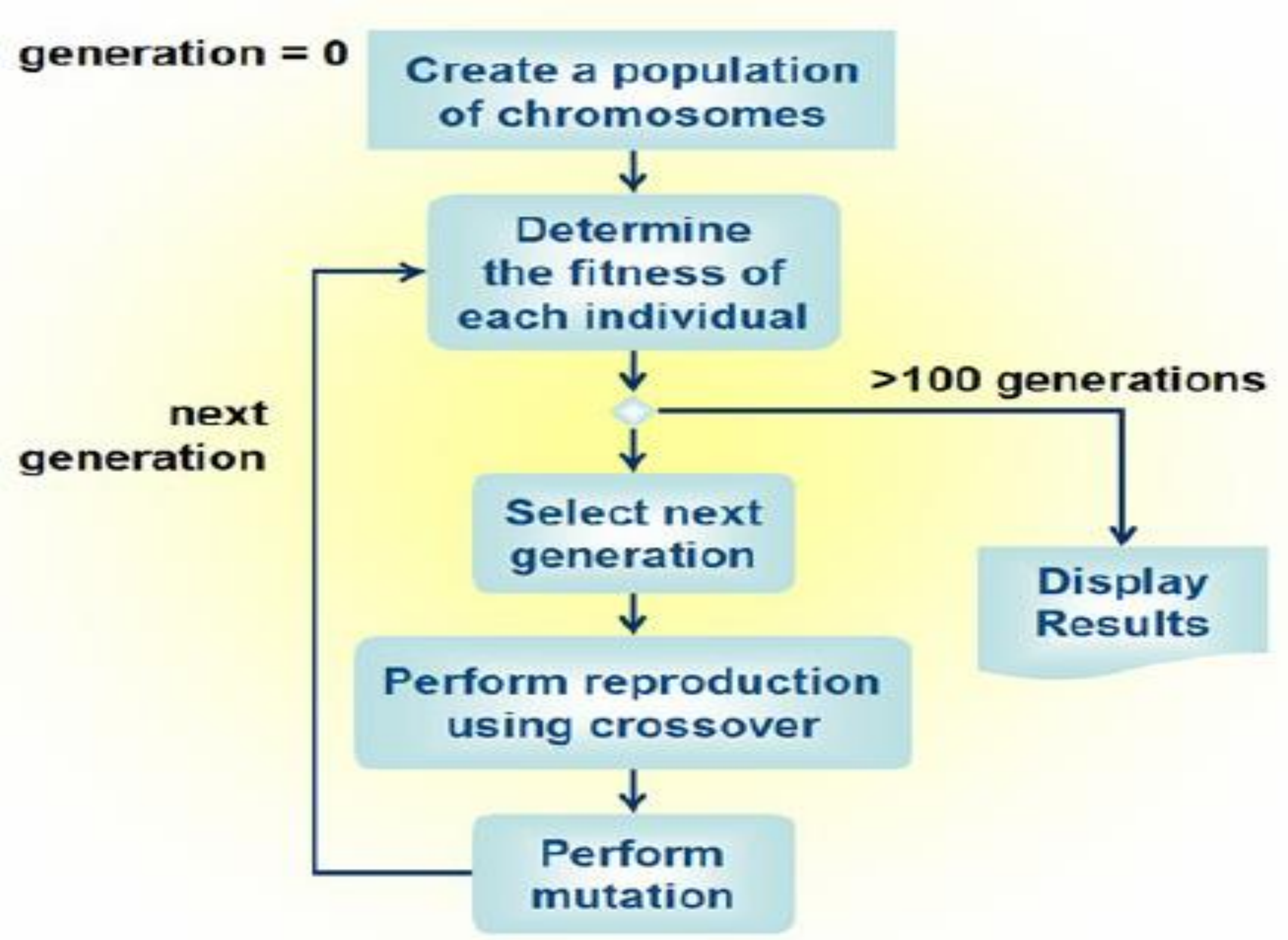


- **Optimization** Technique
- Darwin's **principle of evolution** (**survival of the fittest**) has made this optimization algorithm effective.
- Work with a coding of the parameter set
- Search from a **population of points**, not a single point
- Use payoff (**objective function**) information, not derivatives or auxiliary knowledge
- Use probabilistic transition rules, not deterministic rules

GAs vs. Nature



A solution (phenotype)	Individual
Representation of a solution (genotype)	Chromosome
Components of the representation	Genes
Set of solutions	Population
Selection:survival of the fittest	Darwin’s Theory
Search operators	Crossover and Mutation


generation = 0
Create a population of chromosomes
Determine the fitness of each individual
>100 generations
next generation
Select next generation
Display Results
Perform reproduction using crossover
Perform mutation

Outline of the Basic Genetic Algorithm

- [**Start**] Generate random population of *n* chromosomes (suitable solutions for the problem).
- [**Fitness**] Evaluate the fitness *f(x)* of each chromosome *x* in the population.
- Repeat until terminating condition is satisfied
  - [**Selection**] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
  - [**Crossover**] Crossover the parents to form new offsprings (children). If no crossover was performed, offspring is the exact copy of parents.
  - [**Mutation**] Mutate new offspring at selected position(s) in chromosome).
  - [**Accepting**] Generate new population by placing new offsprings.
- Return the best solution in current population

Issues involved

- How to create chromosomes and what type of encoding to choose?
- How to perform Crossover and Mutation, the two basic operators of GA?
- How to select parents for crossover?

Termination of Loop

- Reaching some (known/hoped for) fitness.
- Reaching some maximum allowed number of generations.
- Reaching some minimum level of diversity.
- Reaching some specified number of generations without fitness improvement.

Advantages and Disadvantages of GA

- Applicable when little knowledge is encoded in the system.
- Effective way of finding a reasonable solution to a complex problem quickly.
- NP-complete problems can be solved in efficient way.
- Parallelism and easy implementation is an advantage.
- However, they give very poor performance on some problems as might be expected from knowledge-poor approaches.

Criteria for GA Approaches

- **Completeness:** Any solution should have its encoding
- **Non redundancy:** Codes and solutions should correspond one to one
- **Soundness:** Any code (produced by genetic operators) should have its corresponding solution
- **Characteristic perseverance**: Offspring should inherit useful characteristics from parents.

Contd…

- The following questions need to be answered:
  - How to create chromosomes and what type of encoding to choose?
  - How to perform Crossover and Mutation, the two basic operators of GA?
  - How to select parents for crossover?
- **Representation of GA :** Binary strings
- **Recombination operator :** N-point or uniform
- **Mutation operator** :Bitwise bit-flipping with fixed probability
- **Parent selection:** Fitness-Proportionate
- **Survivor selection:** All children replace parents
- Emphasis on crossover
- **Speciality:**

Encoding of a Chromosome

- A chromosome should contain information about solution that it represents.
- The commonly used way of encoding is a binary string.

  Chromosome 1: 1101100100110110
  Chromosome 2: 1101111000011110

- Each bit in the string represents some characteristics of the solution.
- There are many other ways of encoding. The encoding depends mainly on the problem.

Encoding and Population

- Encoding: A chromosome encodes a solution in the search space
  - Usually binary string of 0's and 1's
  - Each bit in the string can represent some characteristics of the solution
  - Chromosome size depends on parameter set to be encoded

- Population:
  - A set of chromosomes in a population
  - Population size is usually constant
  - Common practice is to choose the initial population randomly.

Chromosome Size

Chromosome: A set of alleles/genes

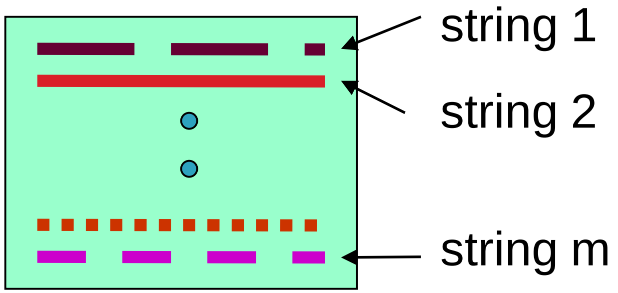
A chromosome is an encoded form of all parameters describing the given problem

An encoded parameter – A subset of alleles

Chromosome size = ??

Population

A population



Evaluation (Fitness) Function

- Represents the requirements that the population should adapt to
  - *quality* function or *objective* function
- The fitness is calculated by first decoding the chromosome and then the evaluating the objective function.
- Fitness function is and indicator of how close the chromosome is to the optimal solution
- Typically we talk about fitness being maximised
  - Some problems may be best posed as minimisation problems, but conversion is trivial

Fitness Evaluation

- Fitness (objective function) associated with each chromosome
- Indicates the degree of goodness of the encoded solution (chromosome)
- only problem specific information (also known as the payoff information) that GAs use
- for minimization problem

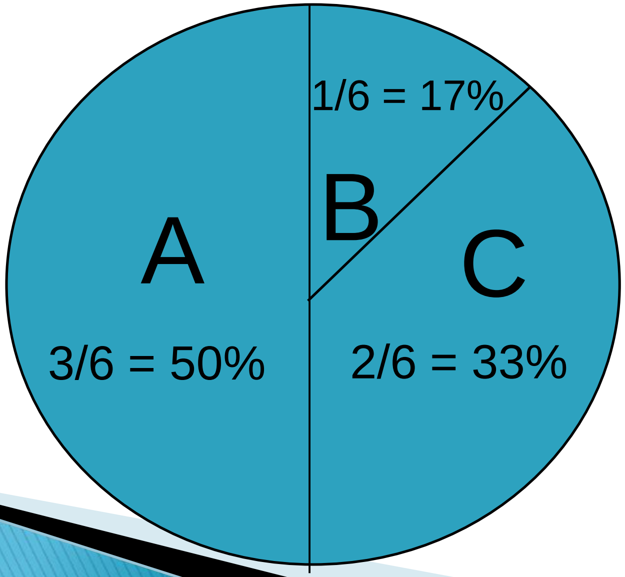
$$\text{fitness} \propto 1/\text{objective}$$

Selection (Reproduction)

- A process in which individual strings are copied according to their objective function or fitness values : Darwin's *survival of fittest*
- Better individuals get higher chance
  - more copies to good strings
  - fewer copies to bad strings
- Mimics the natural selection procedure to some extent
- Implementation:
  - Proportional selection :
  - Roulette wheel selection
  - Tournament selection etc.

Roulette Wheel Selection

- 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

Parent Selection Mechanism

- Depending on their finesses -Assigns variable probabilities of individuals acting as parents
- Usually probabilistic
  - high quality solutions more likely to become parents than low quality
  - but not guaranteed
  - even the worst in current population usually has non-zero probability of becoming a parent
- This *stochastic* nature can aid escape from local optima

Survivor Selection–Replacement

- Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation
- Often deterministic
  - Fitness based : e.g., rank parents + offspring and take best
  - Age based: make as many offspring as parents and delete all parents
- Sometimes do combination of above two

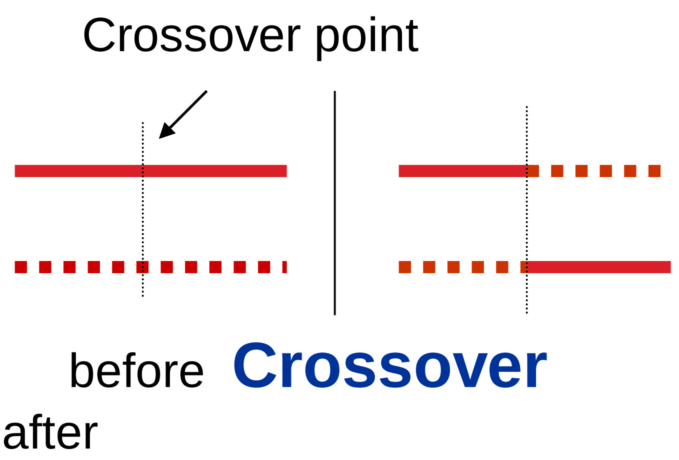
Elitist Model of GAs

The best string up to the current generation is preserved

Crossover

- Partial exchange of information between two randomly selected parent chromosome
- Single point crossover : most commonly used
- Probabilistic operation, μ_c

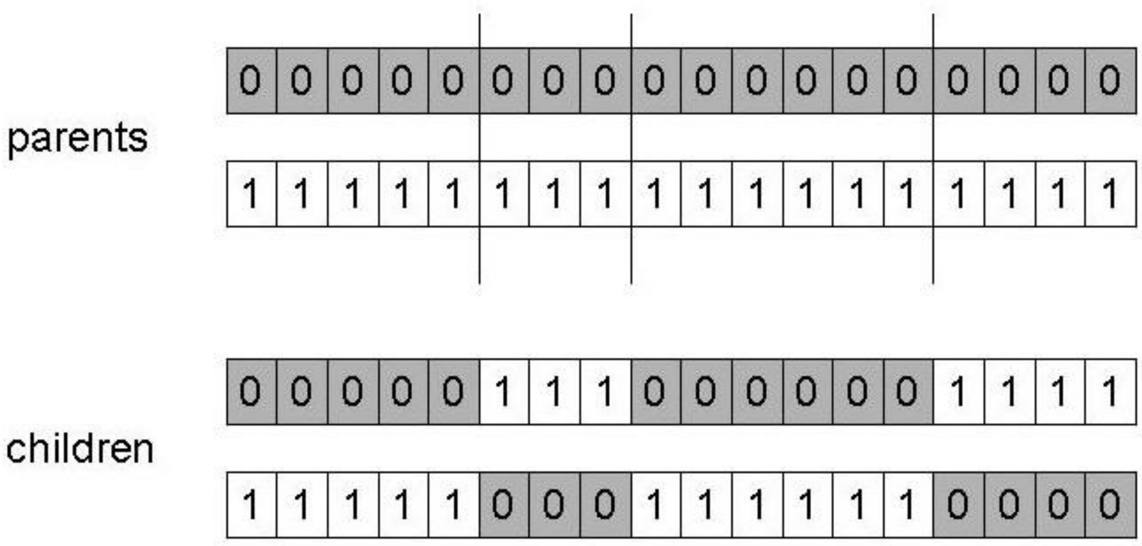
Single point Crossover (example)



Crossover

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)



Crossover

- Crossover operates on selected genes from parent chromosomes and creates new offspring.
- The simplest way is to choose some crossover point randomly
  - copy everything before this point from the first parent and then copy everything after the crossover point from the other parent.

Contd…

- Example: (| is the crossover point):

Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

- There are other ways to make crossover, for example we can choose more crossover points.
- Crossover can be quite complicated and depends mainly on the encoding of chromosomes.
- Specific crossover made for a specific problem can improve performance of the genetic algorithm.

Mutation

- Mutation operation randomly changes the offspring resulted from crossover.
- Mutation is intended to prevent falling of all solutions in the population into a local optimum of the problem.
- In case of binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1.
- Mutation can be then illustrated as follows:

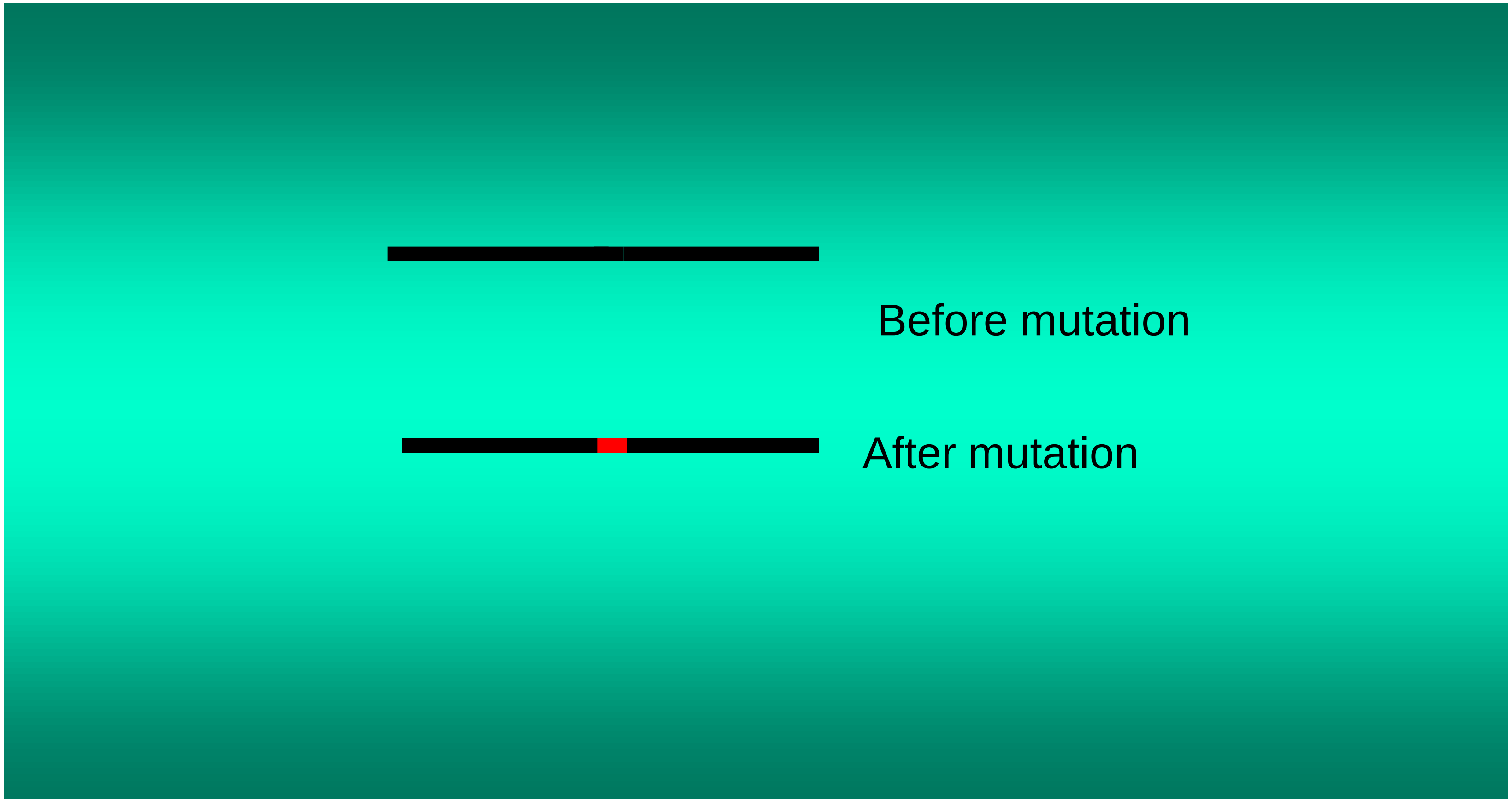
Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110100

- The technique of mutation (as well as crossover) depends mainly on the encoding of chromosomes.

Mutation

- Random alteration in the genetic structure
- Mutating a binary gene : simple negation of the bit
- Probabilistic operation : alter each gene independently with a probability p_m
- Exploring the search area : new search points
- Introduce genetic diversity into the population

Mutation (example)



Crossover and Mutation Schemes

- As already mentioned, crossover and mutation are two basic operations of GA.
- Performance of GA depends on the encoding and also on the problem.
- There are several encoding schemes to perform crossover and mutation.

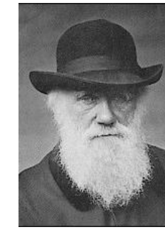
Exploitation vs. Exploration

- **Exploration:** Discovering promising areas in the search space, i.e. gaining information on the problem – new search points
- **Exploitation:** Optimizing within a promising area, i.e. using information - Exploit historical information: survival of the fittest

  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 as well as explorative, it creates random *small* diversions, thereby staying near (in the area of) the parent
- Selection is totally exploitive

An example of SGA (Goldberg)



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

We show one generational cycle done by hand

x^2 example (contd.) : *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 (contd.) : *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 (contd.) : *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

Advantages and Disadvantages of GA

- Applicable when little knowledge is encoded in the system.
- Effective way of finding a reasonable solution to a complex problem quickly.
- NP-complete problems can be solved in efficient way.
- Parallelism and easy implementation is an advantage.
- However, they give very poor performance on some problems as might be expected from knowledge-poor approaches.

Contd..

- There are NP-complete problems that can not be solved algorithmically in efficient way.
- NP stands for nondeterministic polynomial and it means that it is possible to guess the solution and then check it in polynomial time.
- If we have some mechanism to guess a solution, then we would be able to find a solution in some reasonable or polynomial time .
- The characteristic for NP-problems is that algorithm is usually $O(2^n)$ and it is not usable when n is large.
- For such problems, GA works well.
- But the disadvantage of GAs is in their computational time.

- They can be slower than some other methods.
- Some of the problems are listed below
  - Choosing encoding and fitness function can be difficult.
  - GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum in many problems..
- GAs cannot effectively solve problems in which the only fitness measure is right/wrong, as there is no way to converge on the solution.
- In these cases, a random search may find a solution as quickly as a GA.

Shortcomings of SGA

- Representation is too restrictive
- Mutation & crossovers only applicable for bit-string & integer representations
- Selection mechanism sensitive for converging populations with close fitness values

Parameters

- Population size : usually fixed
- String/chromosome length : usually fixed
- Crossover probability : (μ_c)
- Mutation probability : (μ_m)

$$\mu_c >> \mu_m$$

- Termination criteria :
  - desired fitness, if possible (known)
  - generally a maximum number of iterations

Termination Criterion

The cycle of GA-operators (selection, crossover and mutation) is repeated a number of times till :

- A desired objective function value is attained in any string/chromosome of the population

  or

- The average fitness value of a population becomes more or less constant over a specified number of generations

  or

- **The number of generations/iterations is greater than some pre-specified value.**

GA Applications

- Control
- Design
- Scheduling
- Robotics
- Machine Learning
- Signal Processing
- Game Playing
- Combinatorial Optimization

More Specific Applications of GA

- TSP and sequence scheduling
- Finding shape of protein molecules
- Strategy planning
- Nonlinear dynamical systems – predicting, data analysis
- Designing neural networks, both architecture and weights
- Evolving LISP programs (genetic programming)

Discussion

??

Thanks !