Soft Computing Paradigm

Genetic Algorithm

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

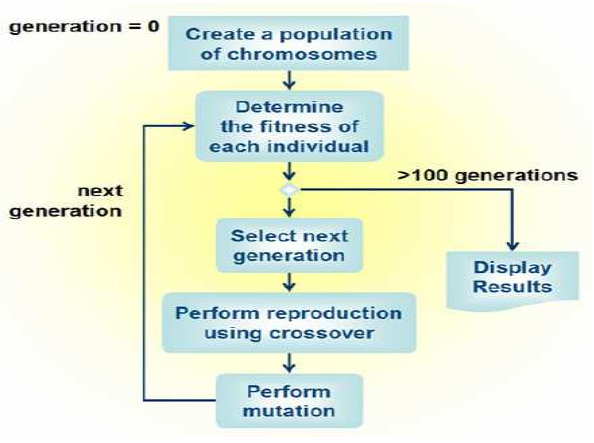
- Conceptual Simplicity
- Broad Applicability
- Hybridization with Other Methods
- Parallelism
- Robust to Dynamic Changes
- Solves Problems that have no Solutions

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.



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:**

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

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

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.

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.

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.

Binary Encoding Schemes

Binary Encoding

1. Crossover

- **Single point crossover**:
  - one crossover point is selected,
  - binary string from the beginning of the chromosome to the crossover point is copied from the first parent,
  - the rest is copied from the other parent

11001011+11011**111** = **11001111**

Contd…

- **Two point crossover**:
  - two crossover points are selected,
  - binary string from the beginning of the chromosome to the first crossover point is copied from the first parent,
  - the part from the first to the second crossover point is copied from the other parent and
  - the rest is copied from the first parent again

1100101**01** + 11**011**11 = **11011101**

Contd…

- **Arithmetic crossover**: Arithmetic operation is performed to make a new offspring
  11001011 + 11011111 = 11001001 (AND)
- **Uniform crossover**: bits are randomly copied from the first and second parent
  11101010 + 11010101 = 11010011

2. **Mutation**

Bit inversion: selected bits are inverted
11010011 => 11110010

Permutation Encoding

1. **Crossover : Single point crossover -**
   - one crossover point is selected,
   - the genes are copied from the first parent till the crossover point, then
   - the other parent is scanned and if the gene is not yet copied in the offspring, it is added
   - *Note: there are more ways to produce the rest after crossover point*

   (1 2 3 4 5 6 7 8 9) + (4 5 3 6 8 9 7 2 1) =
   (1 2 3 4 5 6 8 9 7)
2. **Mutation: Order changing -**
   - two numbers are selected and exchanged

   (1 2 3 4 5 6 8 9 7) => (1 8 3 4 5 6 2 9 7)

Value Encoding

1. **Crossover**

   All crossovers methods from **binary encoding** can be used
2. **Mutation**

   Adding (for real value encoding) - a small number is added to (or subtracted from) selected values

   (1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

Tree Encoding

1. **Crossover**

   Tree crossover - one crossover point is selected in both parents, and the parts below crossover points are exchanged to produce new offspring
2. **Mutation**

   Changing operator, number - selected nodes are changed

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.

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)

Genetic programming

- Genetic programming starts with randomly created computer programs and evolves programs progressively over a series of generations similar to genetic algorithm.
- Furthermore, genetic programming is useful in finding solutions where the variables are constantly changing.
- A population of random *trees* representing programs is constructed.

- The genetic operators (crossover, reproduction, etc.) are performed on these trees.
- In order to create these individuals, two distinct sets are defined:
  - the *terminal set* T, and
  - the *function set* F.
- The terminal set includes variables, as well as constants.
- All the functions and terminals must be compatible (i.e. can pass information between each other).
- Random tree is generated until all the branches end in terminals.
- To generate a population of programs, just generate as many trees as needed.

The steps required in GP

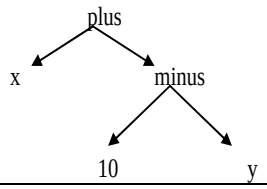
- Initially generate a population of random compositions of the functions and terminals of the problem (computer programs).
- Execute each program in the population and assign it a fitness value according to how well it solves the problem.
- Create a new population of computer programs.
- Copy the best existing programs.
- Create new computer programs by mutation.
- Create new computer programs by crossover.
- The best computer program that appeared in any generation is designated as the result of genetic programming.

Coding Scheme

- In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language.
- LISP programming language is often used for this, as programs in LISP are represented in this form of list and can be easily parsed as a tree.
- The crossover and mutation operations can be done easily.

Chromosome

x + (10 – y) → (plus x (minus 10 y))

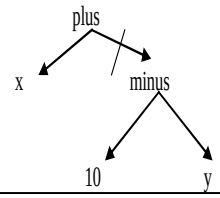


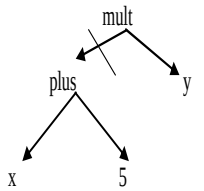
Fitness Function

- The most difficult and important concept of GP is the fitness function.
- The fitness function determines how well a program is able to solve the problem.
- It varies greatly from one type of program to the next.
- For example, if one were to create a genetic program to set the time of a clock, the fitness function would simply be the amount of time that the clock is wrong.

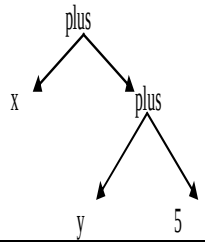
Crossover Operation on two parents

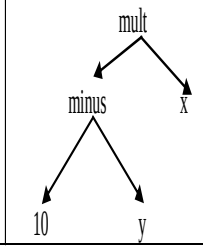
Chromosome A	Chromosome B
x + (10 – y) → (plus x (minus 10 y))	(x + 5) * y → (mult (plus x 5) y)



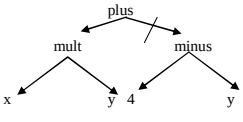
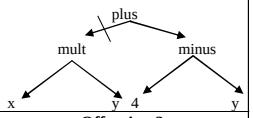
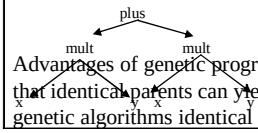
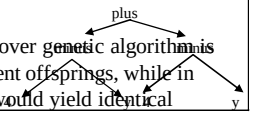


Offspring 1	Offspring 2
x + (y + 5) → (plus x (plus y 5))	(10 – y) * x → (mult (minus 10 y) x)





Crossover Operation with identical parents

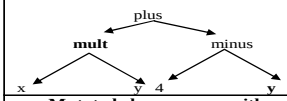
Chromosome A	Chromosome B
x * y + (4 – y) → (plus (mult x y) (minus 4 y))	x * y + (4 – y) → (plus (mult x y) (minus 4 y))
	
Offspring 1	**Offspring 2**
x *y + x * y → (plus (mult x y) (mult x y)))	(4 – y) + (4 – y) → (plus (minus 4 y) (minus 4 y))
	

Advantages of genetic programming over genetic algorithms is that identical parents can yield different offsprings, while in genetic algorithms identical parents would yield identical offspring.

Mutation

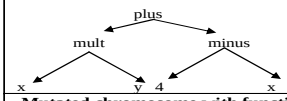
Chromosome

x * y + (4 – y) → (plus (mult x y) (minus 4 y))



Mutated chromosome with variable 'y' replaced by variable 'x'

x * y + (4 – x) → (plus (mult x y) (minus 4 x))



Mutated chromosome with function 'mult' replaced by function 'plus'

x * y + (4 – y) → (plus (mult x y) (minus 4 y))