A Seminar on Genetic Algorithm
by Prakash B. Pimpale & Nitin S. Bhande
Guided by: prof. Suhas S. Gajare
Visit: http://pimpalepatil.googlepages.com/

Thursday, July 02, 2009

Prakash B. Pimpale & Nitin S. Bhande

Department of electronics & telecom. Engg. SGGS Institute of engg. & technology, Nanded.
Contents:

- Introduction
- History
- Biological background
- GAs
- Example
- Applications
- Advantages
- Limitations
- Future scope
- Conclusion
- References
Introduction:

- Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence.

- Genetic algorithms (GA's) are a technique to solve problems which need optimization.

- Based on idea that evolution represents search for optimum solution set.

- GA's are based on Darwin's theory of evolution.
History:

• Evolutionary computing evolved in the 1960's by I. Rechenberg in his work "Evolution strategies"
• GA's were invented by John Holland in the mid-70's.
• John Holland wrote "Adaption in Natural and Artificial Systems" which was published in 1975.
• 1992 John Koza used GA's for Genetic Programming (GP)
Biological background:

- **Chromosome**: string of DNA
- **Genes**: blocks of DNA, responsible for the particular characteristic of the individual
- **Reproduction**: generation of the offspring from parents
- **Crossover**: Genes from parents combine to form a whole new chromosome
- **Mutation**: A bit change in DNA of offspring caused due to error in copying genes from parents
- **Survival of the fittest**: Those who are fit to the environment, survive for the evolution
- **Evolution**: A process giving more & more fit individuals
Genetic algorithms:

- Search space
- Basic algorithm
- Encoding
- Crossover & Mutation
- Selection on the basis of fitness
Search space:

-“The set of solutions among which the desired solution resides”.

-With GA we look for the best solution among among a number of possible solutions - represented by one point in the search space
Basic algorithm:

0 START : Create random population of n chromosomes
1 FITNESS : Evaluate fitness f(x) of each chromosome in the population
2 NEW POPULATION:
   0 SELECTION : Based on f(x)
   1 RECOMBINATION : Cross-over chromosomes
   2 MUTATION : Mutate chromosomes
   3 ACCEPTATION : Reject or accept new one
3 REPLACE : Replace old with new population: the new generation
4 TEST : Test problem criterium
5 LOOP : Continue step 1 - 4 until criterium is satisfied
Pictorial basic algo.

1. **START**
2. Create initial, random population of organisms (potential solutions)
3. Evaluate fitness for each organism
4. 
   - **YES** Optimal or "good" solution found?
     - **YES** END
     - **NO** Reproduce and kill organisms
6. Mutate organisms
**Encoding:**

- “Coding of the population for evolution process”.

**Binary Encoding:** In binary encoding, every chromosome is a string of bits - 0 or 1.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>101100101100101011100101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>111111100000110000111111</td>
</tr>
</tbody>
</table>

**Permutation Encoding:** In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>1 5 3 2 6 4 7 9 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>8 5 6 7 2 3 1 4 9</td>
</tr>
</tbody>
</table>
*Value Encoding:* Values can be anything connected to problem.

<table>
<thead>
<tr>
<th>Chromosome A</th>
<th>1.2324 5.3243 0.4556 2.3293 2.4545</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome B</td>
<td>ABDJEIFJDHDIRJFDLDFLFGT</td>
</tr>
<tr>
<td>Chromosome C</td>
<td>(back), (back), (right), (forward), (left)</td>
</tr>
</tbody>
</table>

*Tree Encoding:* In this every chromosome is tree of some function or command in prog. language

Chromosome A:
```
+      
/ \    
x  j    
5  y
```

Chromosome B:
```
do_until
       
step
     /   
    wall
```

Thursday, July 02, 2009
(Prakash B. Pimpale & Nitin S.)
Crossover:

1. **Single point crossover**: \(11001011 + 11011111 = 11001111\)

   - **Parent A**: \[\text{Blue} + \text{Green}\]
   - **Parent B**: \[\text{Yellow} + \text{Pink}\]
   - **Offspring**: \[\text{Green} + \text{Pink}\]

2. **Two point crossover**: \(11001011 + 11011111 = 11011111\)

   - **Parent A**: \[\text{Blue} + \text{Green} + \text{Blue}\]
   - **Parent B**: \[\text{Yellow} + \text{Pink} + \text{Pink}\]
   - **Offspring**: \[\text{Green} + \text{Pink} + \text{Pink}\]

3. **Uniform crossover**: \(11001011 + 11011101 = 11011111\)

   - **Parent A**: \[\text{Green} + \text{Green} + \text{Green}\]
   - **Parent B**: \[\text{Yellow} + \text{Pink} + \text{Pink} + \text{Pink}\]
   - **Offspring**: \[\text{Green} + \text{Pink} + \text{Pink} + \text{Pink}\]
**Mutation:**

**Bit inversion:** Selected bits are inverted

![Before and after crossover](image)

11001001 => 10001001

**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)
Selection on the basis of fitness:

1. **Rank selection**: Ranks the population first and then every chromosome receives fitness value determined by this ranking. The worst will have the fitness 1, the second worst 2 etc. and the best will have fitness \(N\) (number of chromosomes in population).

Situation before ranking (graph of fitnesses)

Situation after ranking (graph of order numbers)
2. **Roulette Wheel Selection:**

- The size of the section in the roulette wheel is proportional to the value of the fitness function of every chromosome - the bigger the value is, the larger the section is.

- A marble is thrown in the roulette wheel and the chromosome where it stops is selected.
3. Steady-State Selection:

- Allows big part of chromosomes to survive to next generation.
- Few good (with higher fitness) chromosomes are selected for creating new offspring.
- Then some bad (with lower fitness) chromosomes are removed and the new offspring is placed in their place.
- Thus process continues until we get optimal solution.
4. Elitism:
- The best chromosomes are copied first & then the evolution proceeds
- This prevents loss of the best found solution
- Elitism can rapidly increase the performance of GA
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

Graph of cities
Coding for eight city problem:

Encoding used Permutation Encoding:

1) Aurangabad  2) Jalana  3) Parbhani
4) Manvat
5) Nanded  6) Latur  7) Hingoli  8) Selu

CityList1 (1 2 8 7 5 6 4 3)
CityList2 (2 8 7 1 3 4 5 6)
Crossover-

Parent1 (1 2 8 7 5 6 4 3)
parent2 (2 8 7 1 3 4 5 6)

Child (1 2 8 7 3 4 5 6)
Mutation

Mutation involves reordering of the list:

\[
\begin{array}{c c c c c c c c}
& 1 & 2 & 8 & 7 & 3 & 4 & 5 & 6 \\
\end{array}
\]

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

After: \((1 \ 2 \ 8 \ 4 \ 3 \ 7 \ 5 \ 6)\) \(\rightarrow\) this is what the shortest path covering all cities!

- so solution is as 

\[
\begin{array}{c c c c c c c c}
1 & 2 & 8 & 4 & 3 & 7 & 5 & 6 \\
\end{array}
\]
Applications:

- Designing neural networks, both architecture and weights
- In Strategy planning
- TSP and sequence scheduling
- In game playing
- Stock market predictions
- In mathematics
- And many more
Advantages:

- Concept is easy to understand
- Genetic algorithms are intrinsically parallel.
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed
- Less time required for some special applications
- Chances of getting optimal solution are more
  and many more
Limitations:

- The population considered for the evolution should be moderate or suitable one for the problem (normally 20-30 or 50-100)
- Crossover rate should be 80%-95%
- Mutation rate should be low i.e. 0.5%-1% assumed as best
- The method of selection should be appropriate
- Writing of fitness function must be accurate
Future scope:

• For finding new crossover methods for more convergence in single step
• For finding new mutation methods for reducing divergence due to excess mutation
• For finding a function which will decide suitable number of initial population
Conclusion:
- Genetic algorithms are rich - rich in application across a large and growing number of disciplines.
- Really genetic algorithm changes the way we do computer programming.
References:

2) *Genetic Algorithms and Evolutionary Computation*
   by Adam Marczyk
4) Paper on *Recent development in evolutionary & genetic algorithm: theory & applications.*
   by N.Chaiyaratana & A.M.S.Zalzala
5) [www.gwnetic~programming.com](http://www.gwnetic~programming.com)
Questions?