Application of Genetic Algorithms for Optimization

Atulya Shivam Shree(10327172), Avinash Bhattarmakki(10175), Vikas Kumar Singh(10801)

Term Paper: EE 698W

Jan, 2014 - Apr,2014

***Abstract*—Genetic Algorithms are a class of search heuristic problems which replicate the biological evolutionary mechanism in order to solve a number of problems. These algorithms learn with their experiences and adapt to change according to the environment, hence also know as evolutionary algorithms. In this project we identify some interesting problems in Graph Theory and solve them using these Evolutionary Algorithms.**

I. Introduction

GENETIC algorithms are a class of methods which are used for stochastic optimization of a problem. The fundamental logic behind these algorithms is that an evolutionary process is started which tries to maximize the fitness of a process by learning and adapting to the new environment. The algorithm can be easily implemented on a parallel computational architecture. Also the inherent design of the process does not allow it to get stuck in a local minima.
Genetic algorithms are useful for all those problems in which there is not enough information to build a differentiable function or where the problem has such a complex structure that it is difficult to conceptualize the influence of different variables on the target function

Graph Theory deals with the study of graphs which are mathematical structures to model pair-wise relationship between objects. A Graph is made up of vertices or nodes which represents the objects and edges which represent the link between the connecting objects.

A simple implementation of the graph is in the form of $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. Here V is a set of vertices while E is a set of edges connecting the vertices V. Graph Theory has widespread application in Computer Science, study of molecules in Chemistry. If a graph represents a map of a given area there would be number of problems of finding optimal paths between two nodes within the network.

In this project we first explore the techniques used in GA for optimization problems. A simple implementation of solving a non-convex problem has been shown to emphasize the advantage of using GA. After this we deal with two problems in graph theory which are computationally expensive to solve. Through these examples we demonstrate how GAs can be used to obtain a faster solution in comparison to other techniques.

This report is divided into 5 sections : Section II deals with the conceptual structure of Genetic Algorithms(henceforth abbreviated as GA). Section III is on application GA in solving a non-convex problem. Results of optimality and influence of techniques have been discussed there. In Section IV we pick up the problem of finding an optimal path in a graph and implement it using GA. Section V looks at the NP-complete problem, namely coloring of graph is implemented using GA. The objective here is to color all nodes of a graph using minimum number of colours provided that the adjacent nodes do not have the same colour.

II. Structure of Conventional Genetic algorithms

The most common techniques used in genetic algorithms are as follows:

- Encoding
- Selection
- Recombination
- Mutation

A. *Encoding the Genes*

The *coding* of the optimization problem produces the required discretization of the variable values (for optimization of real functions) and makes their simple management in a population of search points possible. Normally the maximum number of search points, i.e., the *population size,* is fixed at the beginning. The encoded form of a variable is termed a gene. A search point is the combination of all the genes for the given problem and is termed a chromosome.

B. *Selection and Fitness*

The optimization objective is identified in form of a fitness function. Depending on the encoding scheme we try to find out the fitness score of each member of the population. After each generation fitness score of the population is generated.

The rule of the survival of the fittest takes place here and only the healthy individuals are passed on for the next generation.

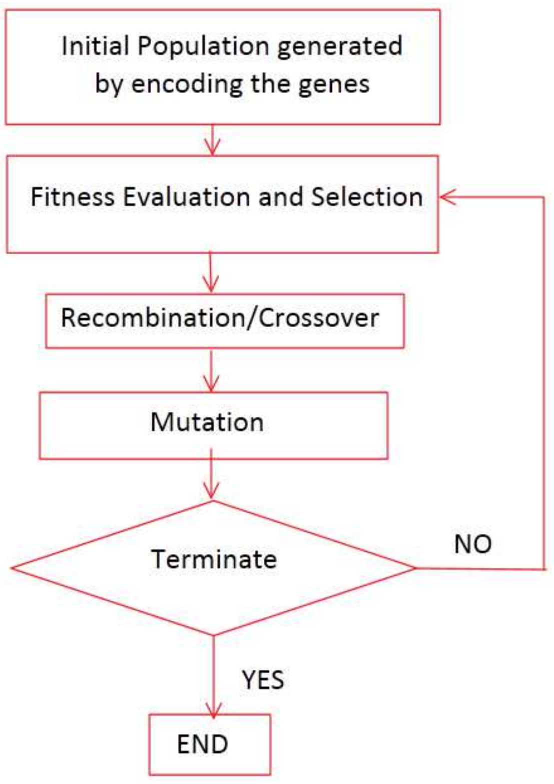
C. Recombination

The recombination or information exchange operators allow new population generated to contain information from multiple chromosomes of the previous generation. This allows the optimization process to improve the search space after every step. The basic process is that two chromosomes A and B combine to produce a child C. A crossover point for the chromsome is defined. All the genes to the left of this point come from A while the remaining come from B. This is simple reconstruction of the natural reproduction process in nature.

D. Mutation

The mutation operator decides the probability with which the chromosomes are modified. In a stochastic process this is the main function at play, however in GA it takes place only when recombination takes place. the importance of mutaion stems from the fact that a random change in a gene can allow the chromosome to escape from the trap of a local minimum and explore the local neighborhood for solutions.

The overall execution of the algorithm is described in the following flowchart.


Initial Population generated by encoding the genes
Fitness Evaluation and Selection
Recombination/Crossover
Mutation
Terminate
NO
YES
END

III. Optimization of the Rastrigin function

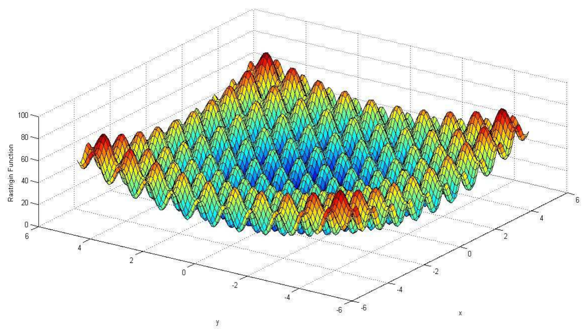
Rastrigin function is a non-convex function used for performance analysis of optimization algorithms. It is a typical example of non-linear multimodal function. Finding the minima of this function is a difficult task since it has large number of local minima and a large search space.

The Rastrigin function is defined by

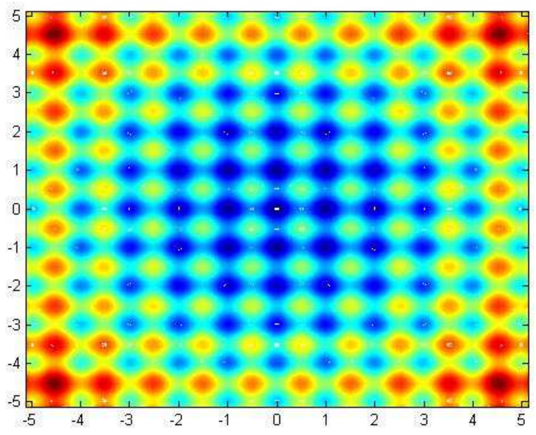
$$f(\mathrm{x}) = An + \sum_{i=0}^{n}[x_i{}^2 - A\cos(2\pi \mathrm{x_i})]$$

Where A = 10 and $x_i \in [-5.12\,, 5.12]$. It has a global minima at $\mathrm{x} = 0$ where $f(\mathrm{x}) = 0$.

A two dimensional Rastrigin function was used to check the functionality of genetic algorithm.



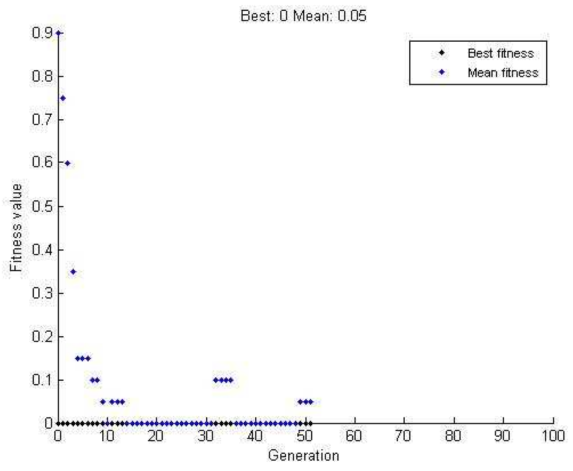
Surface plot of the 2-D Rastrigin function



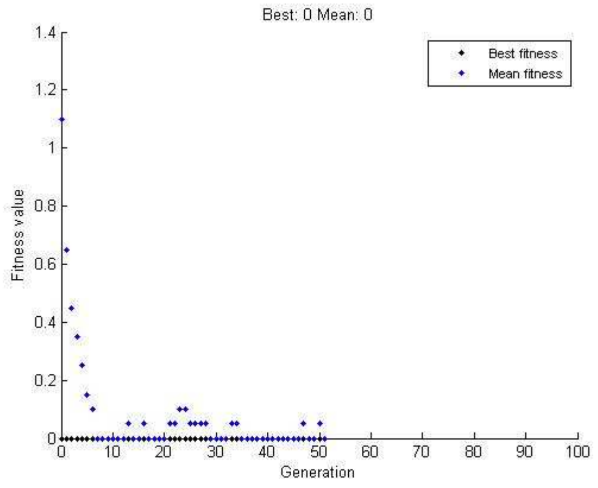
Contour plot of the 2-D Rastrigin Function

To optimize this function the Genetic Algorithm tool of Matlab was used. It was success fully able to find the global minima. The whole 2-D space was divided into 100 X 100 points and each point was then represented in binary. Out of all the points 20 points was chosen randomly with uniform probability. These 20 points were the starting generation of the population. The binary representation was used do represent each digit as a chromosome. The ranking of each entity was decided by the value of Rastrigin function at that point lower the value higher the ranking. Then in each iterations crossover and mutation was done leading the formation of new population the poor ranked entities was then dropped out.

The algorithm was able to find the global minima in about 10 iterations with best fitness value = 0 and mean fitness value ≤ 0.5



Fitness value vs no. of generations

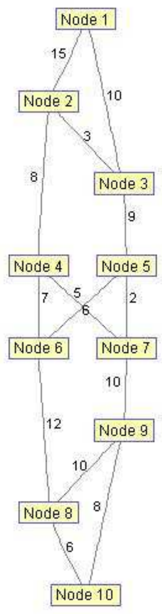


Fitness value vs no. of generations

IV. Finding shortest path in a Graph

Genetic algorithms are generally not used for solving graph search problems as many crossover and mutation operators are not physically meaningful. However, the genetic algorithm approach has the potential to perform well on more general graph search problems and without the effort of handcoding a different set of heuristics for each problem. Genetic Algorithm may offer significant benefits over more typical search of optimization techniques. One of the disadvantages with A* and hierarchical search is that both algorithms are typically deterministic, depending on the exact heuristics used. Thus, for a given problem they will always return the same answer. Since the genetic algorithm approach is not based on any specific heuristic and is randomized, it will not be as vulnerable to this problem.

A fixed network of 10 nodes is used. The optimum path and optimum cost is first calculated using Dijkstra's algorithm using MATLAB command *getshortestpath.*



We get the solution as

```
  cost =
     39
  path =
      1    3    5    7    9    10
  pred =
      0    1    1    2    3    5    5    6    7    9
```

Genetic Algorithm Setup

A. Initial Population-

Initial population(chromosomes) consists of set of paths from start node to end node. We have used any path from the source node to the destination node is a feasible solution. Initial population is created by fixing start_node and end_node, and using Random set of nodes from 2 to 9. One feasible solution (path-1-2-3-5-7-9-8-10) is included in the initial population so that the solution converges fast. The initial population is then encoded using binary representation of the node number. Since the network has 10 nodes, each node is coded in 4 binary units. Hence length of each chromosome = 10x4 = 40 bits.

B. Evaluation-

The population is evaluated using fitness function. The fitness function used here is

$$F = \begin{cases} \dfrac{1}{\sum_{i=1}^{N-1} C_i(g_i g_{i+1})}; & Feasible\ solution \\ 0\,; & Infeasible\ solution \end{cases}$$

Where, $C_i(g_i g_{i+1})$ is the cost between gene g_i and adjacent gene g_{i+1} in the chromosome of N genes (nodes).

C. Reproduction and Selection-

During the reproduction cycle, the population is first ranked according to its fitness value. Then the chromosome with the highest fitness value is duplicated and one with lowest fitness value is eliminated.

D. Crossover-

Crossover is applied by randomly selecting two parent chromosomes and cutting it randomly at two points and exchanging the middle part to create two new child chromosomes. If child is a feasible solution, then child will replace parent in next generation, else parent will prevail. During crossover, start_node and end_node are not affected. Crossover is made to occur with probability 0.7.

E. Mutation-

Mutation takes place by flipping of a bit in one of the chromosomes. A random chromosome is chosen and a random bit is flipped in the chromosome. Mutation takes place between bits 5 and 36 so that start_node and end_node are preserved. Mutation occurs with probability 0.8

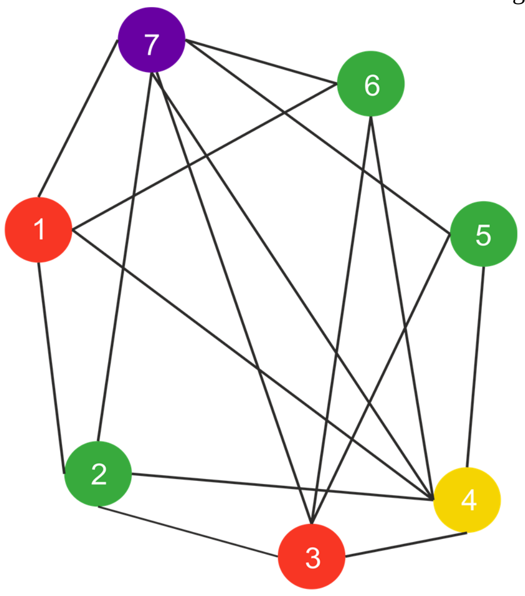
The simulation after 30 generations is given by-

```
39   1   3   5   7   9   10
39   1   3   5   7   9   10
39   1   3   5   7   9   10
39   1   3   5   7   9   10
39   1   3   5   7   9   10
47   1   3   5   7   9    8   10
```

Thus, the solution given by Genetic Algorithm is the path 1-3-5-7-9-10 with the cost of 39, which is same as the solution given by Dijkstra's algorithm.

V. The Graph Coloring Problem

The Graph Coloring problem is a standard NP-Complete problem of the coloring the vertices of a graph. Given a graph the objective is to colour the vertices of a graph such that if two vertices are connected by an edge they must have different colours. The primal objective is to have minimum number of bad edges,i.e., edges where the two vertices connecting them have the same colour. The secondary objective is to reduce the number of colours used in the graph.



1	3	1	2	3	3	4

The practical applications of this problem include :

- Register Allocation
- Pattern Matching
- Radio Frequency Assignment
- Sudoku

The algorithm is tested against the standard DIMACS benchmark tests. For each run the maximum number of colours to be used is fixed and an attempt is made to find a solution to the graph. The minimum number of colours in which we can solve the problem is known as the chromatic index of the graph.

A. Encoding the problem in GA

The chromosome representation of the GA is simply a vector of size equal to the number of vertices and each value of the vertex containing an integer value between 1 and num_colors.

The overall GA we use is the generational GA. The population size for the run is kept constant at all times. After each generation the fitness score of the population is evaluated. The top half healthy population is preserved while new children are generated to replacing the remaining half of the population.

Selecting the fitness function

The fitness score is defined for each of the chromosome in the population. For each chromosome we assume that the graph has been coloured using this strategy and find the number of bad edges present in the graph. A bad edge is defined as one which has both vertieces forming it of the same colour. The fitness score is the negative of the total number of bad edges present in the graph. Hence the ideal fitness of each chromosome is 0 and in general the chromosomes have a negative fitness index.

B. Parent Selection

Two different parent selection strategies have been tested in the implementation. Both were implemented and the first method performed better than the second.

Parent Selection1 :

```
tempParents = select four different chromosomes from the population

parent1 = the fitter of first two of tempParents ;
parent2 = the fitter of last two of tempParents ;

return   parent1, parent2;
```

Parent Selection2 :

```
parent1 = fittest chromosome of the generation
parent2 = fittest chromosome of the generation

return   parent1, parent2;
```

After parents have been selected this the basic recombination strategy using a randomly chosen crossover point is used to generate the children.

C. Mutation:

The implementaion of mutation within a chromosome has been done as follows :

```
mutation_vertices = mutation_index*chromosome size;

for each(vertex in mutation_vertices) {
   if (vertex has the same color as an adjacent vertex) {

   adjacentColors = all adjacent colors;
   validColors = allColors – adjacentColors;

   newColor = random color from validColors;
   chromosome.setColor(vertex, newColor)
   }
}
```

D. Dataset used and Implementation

For this problem we have used the DIMACS benchmarking graph collection. DIMACS stands for Center for Discrete Mathematics and Theoretical Computer Science. It is part of Rutgers University (The State University of New Jersey Rutgers, et al. 2011). The data is commonly used in graph coloring problems. Each of the file consists has a .col extension.

The implementation of GA has been done on Java as it allowed easy integration of graph data structures. The JGRAPHT library has been used for creating the graph data structure. For visualization the mxgraph package of JGRAPH has been used.

E. Results

File	|V|	|E|	Expected Chromatic Score	Minimum colors obtained	Time (s)
myciel3.col	11	20	4	4	0.013
myciel4.col	23	71	5	5	0.018
queen5_5.col	25	320	5	5	0.010
queen6_6.col	36	580	7	7	0.048
myciel5.col	47	236	6	6	0.020
queen7_7.col	49	952	7	9	0.087
queen8_8.col	64	1456	9	11	0.107
huck.col	74	602	11	11	0.020
miles250.col	128	774	8	8	0.049
miles1000.col	128	6432	42	43	0.550
fpsol.i.1.col	496	11654	65	68	>1

Table 1 displays the results of the execution for the DIMACS dataset.

The Columns represent :

- number of vertices |V|
- number of edges |E|
- Actual chromatic score of the graph
- Minimum number of colors that we obtained from our execution
- Execution time

The tests were performed on a laptop computer with the following specifictions:
CPU: Intel Core i7 3612 @2.1Ghz
RAM: 8 GB DDR3 @1333MHz

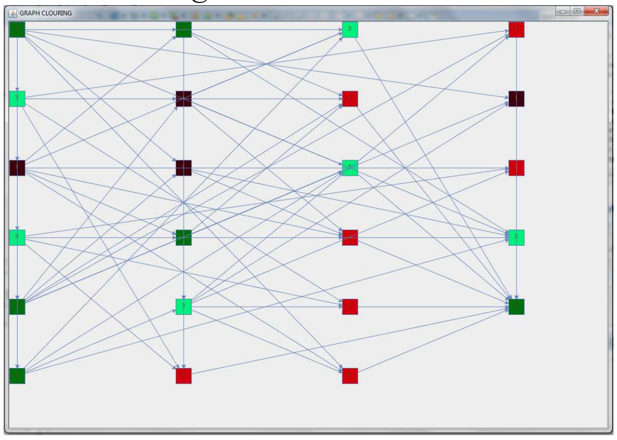
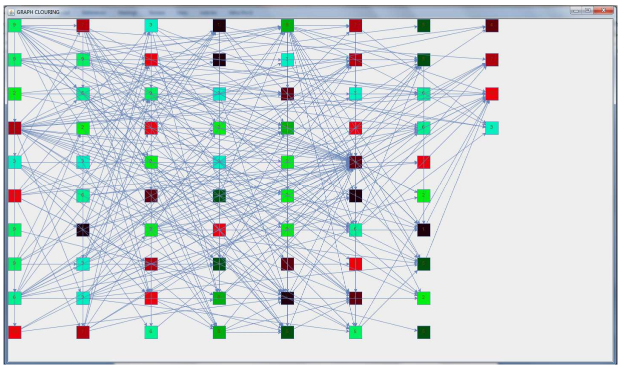


Figure 1 : Results obtained on the myceil4.col file



VI. CONCLUSION

After solving 3 optimization problems using GA we can make the following conclusions:

- GA can solve optimization problems using its heuristic structure
- GA depends on the initial population for the entire run, hence it is necessary to select a diverse
- GA may not always converge to the solution for the same set of parameters in different runs
- Having a large initial population increases the chances of obtaining the optimal solution
- Increasing the mutation index is helpful when the solution is close to optimal and further improvement in fitness is expected
- If we see Table[1] for datasets myciel5 and queen7_7 the nodes are same while the edges have a huge margin. Hence the optimal solution is extremely difficult to obtain for the GA as is evident from the fact that it arrives at the value of 9 for queen7_7 while it easily arrives at 6 for myciel5.

VII. REFERENCES

1. Musa M. Hindi and Roman V. Yampolskiy,*" Genetic Algorithm Applied to the Graph Coloring Problem "*, Proceedings of the Twenty-thirdMidwest Artificial Intelligence and Cognitive Science Conference 2012, p68-74,
2. DIMACS dataset is available on http://mat.gsia.cmu.edu/COLOR/instances/
3. Popov A. ,"*Genetic algorithms for optimization – application in the controller synthesis task*" , diploma thesys, department Systems and Control, faculty Automatics, Technical University ,Sofia, 2003
4. Reza Abbasian and Malek Mouhoub ,"*An Efficient Hierarchical Parallel Genetic Algorithm for Graph Coloring Problem"*, GECCO' 11Proceedings of the 13th annual conference on Genetic and evolutionary computation
5. Gihan Nagib and Wahied G. Ali, "*Network Routing Protocol using Genetic Algorithms*", International Journal of Electrical & Computer Sciences IJECS-IJENS Vol:10 No:02
6. Yagvalkya Sharma, Subhash Chandra Saini, Manisha Bhandhari, "*Comparison of Dijkstra's Shortest Path Algorithm with Genetic Algorithm for Static and Dynamic Routing Network*", International Journal of Electronics and Computer Science Engineering
7. Ben Mowery," *Solving the Generalized Graph Search Problem with Genetic Algorithms*"
8. Subhadip Samanta, "*Genetic Algorithm: An Approach for Optimization (Using MATLAB)"*, International Journal of Latest Trends in Engineering and Technology (IJLTET)