Harnessing Genetic Algorithm for Vertex Cover Problem

Harsh Bhasin
Computergrad.com
i_harsh_bhasin@gmail.com

Geetanjli Ahuja
Amity University, Noida
ahujageetanjli@gmail.com

***Abstract*: The problem of finding a minimum vertex cover is an NP hard optimization problem. Some approximation algorithms for the problem have been proposed but most of them are neither optimal nor complete. The work proposes the use of the theory of natural selection via Genetic Algorithms (GAs) for solving the problem. The proposed work has been tested for some constrained inputs and the results were encouraging. The paper also discusses the application of genetic algorithms to the solution and the requisite analysis. The approach presents a Genetic Algorithms based solution to a problem.**

Keywords- Genetic Algorithms, NP Hard Problem, Artificial Intelligence, Vertex Cover Problem, Non Deterministic Problems.

I. INTRODUCTION

A vertex cover of a graph G is a set C of vertices such that each edge of G is incident to at least one vertex in C. The set C is said to cover the edges of G. A minimum vertex cover is a vertex cover of smallest promising size. The vertex cover number is the size of a minimum vertex cover.

Genetic algorithms (GAs) are a heuristic exploration course which is based on premise of genetic change and endurance of fittest [6]. It is functional when exploration space is too large and the solution can be assigned some fitness value. In Vertex Cover there can be numerous solutions which can be judged by assigning fitness value on the basis of to what extent the cover is minimum. Therefore, the problem is a fit case for applying GAs. The application of GA to the problem can make set of solution enhanced. So before applying GAs there must be a randomized algorithm which solves Vertex Cover. The work presented proposes a randomized algorithm and then improves it by applying genetic.

II. LITERATURE REVIEW

An extensive literature review was carried out. 10 papers on NP hard problems and Vertex cover were studied. In addition to that many sites and books were reviewed. The algorithms proposed were implemented and analyzed. It was found that some approximation algorithms have been given for vertex cover problem but it has been established that approximation algorithms proposed so far neither gives vertex cover in all the cases nor the vertex cover obtain has size less than twice the minimum size of optimal solution. So, the algorithms are, at best, true in some of the cases and not optimal in most of the cases [1, 2].

Since vertex cover problem has many applications including in biology. Therefore the quest to find its optimal solution has driven both computer scientist and researchers in field of medicines and genetics towards this expedition. In one of the papers it has been observed that if the high degree method is incorporated with other pre-processing rules. Then running pre-processing rules before attempting any kernelized method gives the best solution. This method can be used to find pathogenic tree on protein domain [3] [4].

The solutions proposed in various papers suffer from intractability. Therefore, some of the researchers have proposed fixed parameter tractability algorithm that runs in $O(f(k)\, n^c)$ time where c is constant and n is the size of the problem set. The proposed work intends to do away with time constraint by incorporating GAs in the solution.

III. VERTEX COVER PROBLEM

The optimization problems such as minimum spanning tree and vertex cover are NP hard. There is hardly a possibility of finding exact algorithm for the above problems [1]. The importance of vertex cover lies in the fact that it can be used in colossal number of applications. In the problem if G = (V,E) is a graph then we need to

find C such that C is subset of V and C covers all edges in E i.e. every edge ϵ V is incident on at least one vertex in C.
Approximation algorithm has been proposed for the above problem but is open to improvement [7]. The point can be proved by taking the following illustration.
Approximation algorithm for vertex cover is explained below.

```
1. C←φ
2. While E ≠ φ
        Pick any {u, v} Є E
        C ← C ∪ {u, v}
        Delete all edges incident to either u or v.

   Return C
```

To understand the above point a graph shown in figure 1 is taken. First of all, edge 1 is taken and vertices H and A are booked in set C. Initially, C = φ and after taking edge 1, C becomes {H, A}.
All the edges with A or H are deleted. So, the graph will not have edge 1, 2 and 3 in next iteration. Going sequentially, we select edge 4 and take C, D in the set. Thus making C = {H, A, B, C}. Going by same method then we select the edge 6. Thus including vertices {E, D} in the set. Thus, C becomes {A, B, E, D, H} which is surely a vertex cover but not the minimum vertex cover.
The above illustration proves that approximation algorithm does not always produce best result.
The above algorithm has been implemented and the requisite data has been collected and analyzed. The results have been shown in the table below. The value 10 has been assigned to the best solution and the index decreases as the result produces more vertices than minimum set. The results have been summarized in Table I, Table II and Figure 2.
The graph and the tables emphasize the need of a better solution as the above solution fail to produce the best result most of the times. The work presented proposes a method based on the GAs and analyses it. The comparison of the method with the existing solution has also been studied.

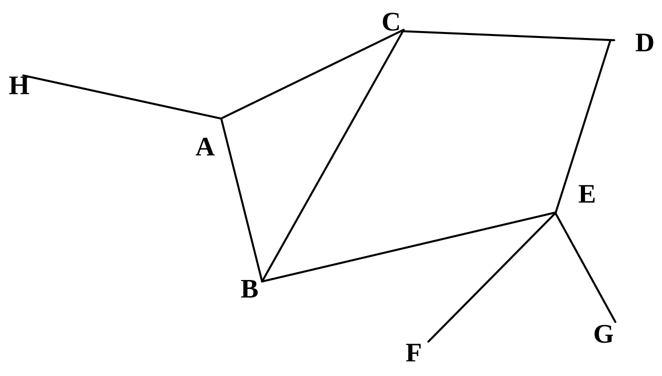

Figure 1: Graph having vertices A, B, C, D, E, F, G and H

TABLE I: OUTPUT OBTAINED BY RUNNING THE PROGRAM FOR THE GRAPH STATED ABOVE AS INPUT

	First Vertex(Randomly picked)	**Output(Vertex Cover)**
RUN 1	C	C,D,E,H,I
RUN 2	B	B,C,A,E
RUN 3	C	C,D,E,B,A
RUN 4	D	D,C,B,A,E
RUN 5	G	G,E,B,A,C
RUN 6	F	F,E,B,C,A
RUN 7	F	F,E,D,C,B,A
RUN 8	A	A,C,D,E

RUN 9	F	F,E,B,A,D
RUN 10	H	H,A,C,B,E
RUN 11	C	C,B,A,E
RUN 12	B	B,E,H,A,C
RUN 13	E	E,B,A,D
RUN 14	D	D,C,A,B,E
RUN 15	H	H,A,B,C,D,E
RUN 16	G	G,E,B,C,A
RUN 17	B	B,E,D,C,A
RUN 18	D	D,E,B,C,A
RUN 19	F	F,E,B,A,C
RUN 20	C	C,D,E,B,A

TABLE II: STATES INDEX OF ALL THE RUN EXECUTED

	Size of vertex cover obtained	**Index**(10 – (Vertex Cover obtained-Minimal Vertex Cover))
RUN 1	5	8
RUN 2	4	9
RUN 3	5	8
RUN 4	5	8
RUN 5	5	8
RUN 6	5	8
RUN 7	6	7
RUN 8	4	9
RUN 9	5	8
RUN 10	5	8
RUN 11	4	9
RUN 12	5	8
RUN 13	4	9
RUN 14	5	8
RUN 15	6	7
RUN 16	5	8
RUN 17	5	8
RUN 18	5	8
RUN 19	5	8
RUN 20	5	8

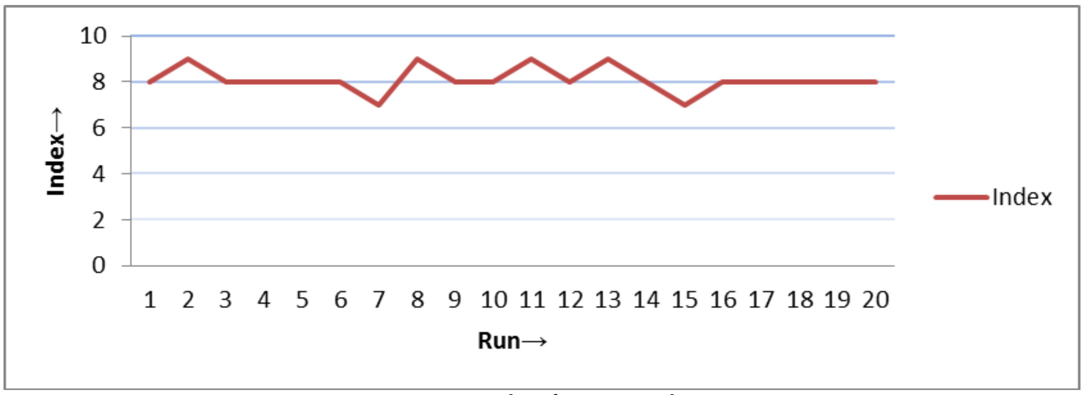

Figure 2: Index of various run during execution

IV. GENETIC ALGORITHMS

Genetic Algorithms (GAs) are search procedures to converge to optimal solution based on the theory of survival of the fittest. The fundamental unit of GA is chromosome. Each chromosome represents a solution to the problem and is composed of a string of cells of finite length. The binary alphabet {0, 1} is often used to represent these cells but integers can be used depending on the application. The fitness value is function or objective against which chromosome is tested for its suitability to the problem in hand.

GAs transforms the population of mathematical object into new population with a better fitness using the following operators:

A. *Crossover*: It is genetic operator that combines two chromosomes to produce a new chromosome. The child chromosome takes one section of the chromosome from each parent. The point at which chromosome is broken depends on the randomly selected crossover point.

   The number of crossover is determined by the crossover rate which is generally 2-5%. GAs have following type of crossover:

   I. *Single point crossover:* In this, a single chromosome point is selected and is applied on two chromosome selected as predicted in figure3.

   II. *Two point crossover:* In this type of crossover two crossover points are selected and the crossover operator is applied as shown in figure 4.

   III. *Uniform Crossover*: In this type bits are copied from both chromosomes uniformly as shown in the figure 5.

B. *Mutation*: It is necessary for ensuring genetic diversity in population. GAs involves string-based modifications to the elements of a candidate solution. These include bit-reversal in bit-string GAs or shuffle and swap operators in permutation GAs (Figure 6).

C. *Selection*: It is quantitative criterion based on fitness value to choose which chromosomes from population will go to reproduce. Intuitively the chromosome with more fitness value will be considered better and in order to implement proportionate random selection Roulette wheel selection is used for selection.

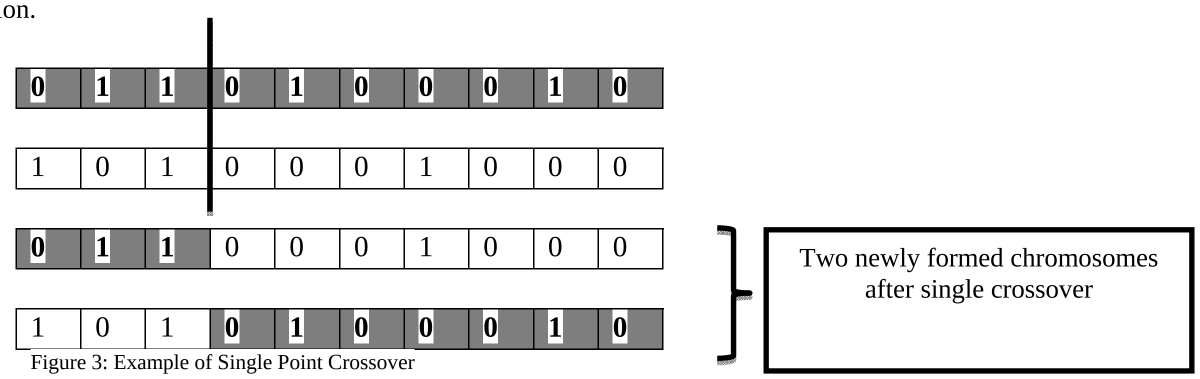

Figure 3: Example of Single Point Crossover

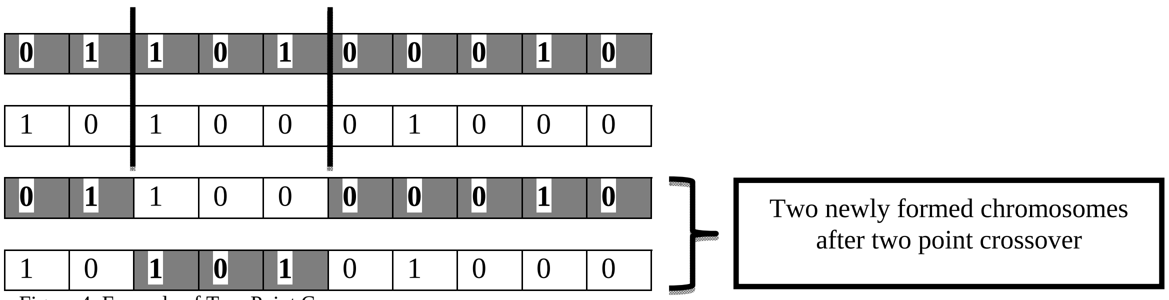

Figure 4: Example of Two Point Crossover

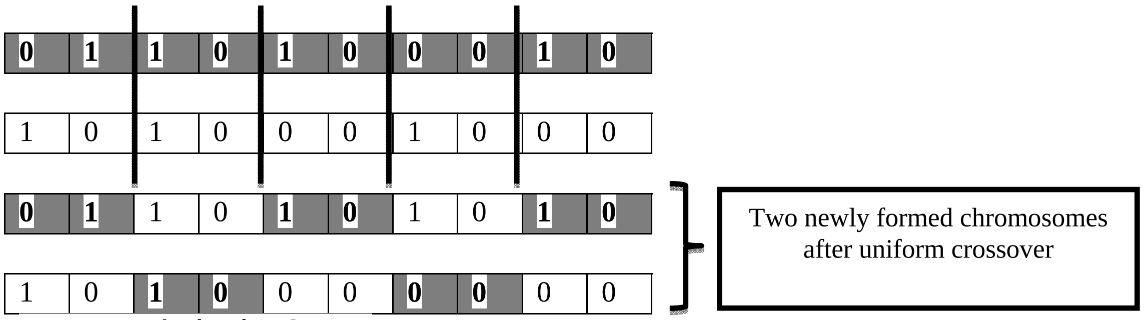

Figure 5: Example of Uniform Crossover

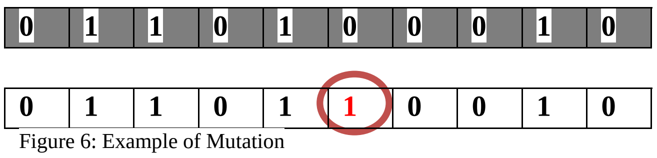

Figure 6: Example of Mutation

V. PROPOSED WORK

Step 1: An initial population consisting of m chromosomes is taken. Each chromosome has n cells where n is the number of vertices in graph.
The population created is binary one where each cell is obtained by taking random number between 0 and 100and checking whether that number is greater than 50 or not. If it is greater than 50 then cell becomes 1 otherwise it becomes 0.
Step 2: Each chromosome consist of 0s and 1s.A particular chromosome represent which vertex to take depending upon whether position is 0 or 1.For e.g. If graph consists of 8 cells then chromosome 01001001 implies that second, fifth and last vertices are to be taken in vertex cover.
Step 3: Each chromosome is assigned an index based on the above step. The index will be more if the number of vertices selected is less. We start the index from 100 and decrease it gradually.
Step 4: For each chromosome fitness value is calculated according to formula $1/(1+e^{-\lambda})$ where λ is vertex cover obtained in step 3.
Step 5: A threshold is decided which is taken half the range of indices obtained in step 3.
Step 6: Crossover and mutation are performed and new chromosomes are subjected to step 3 and 4.
Step 7: Reproduction is carried out with Roulette wheel Selection.
Step 8: The above process can be repeated and the results can be compared with previous iteration in order to improve solution.
The process can be understood with the help of the following figure 7.

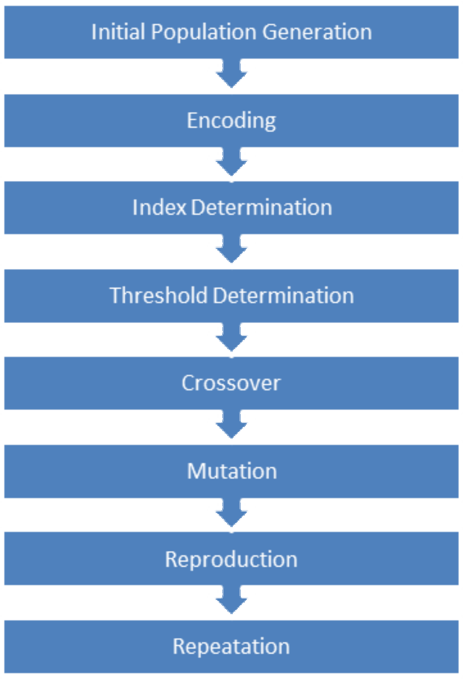

Figure 7: Process of proposed work

VI. RESULTS AND CONCLUSIONS

In the above work, it has been established that, applying the above technique results in the generation of set of solutions that provides almost near if not certain answer to the problem. The Vertex Cover being an NP Hard problem can be solved by algorithms of complexity O (2^n) or O $(n.2^n)$ by the known methods. The method proposed above presents a solution which has complexity far below the known methods. The algorithm has been tested limited number of time due to constraint in resources and time. If the above method undergoes regress testing, there is a high plausibility that it might lead to a fresh, novel approach that uses the blend of nature along with technical novelty to produce encouraging results. This technique can be used to solve other NP hard problems as well.

VII. REFERENCES

[1] Approximation Vertex Cover,cs.dartmouth.edu,cs105, febraury 21,2005.
[2] H.Coreman, Introduction to Algorithms, 3rd ed,Cambridge, Mass.: MIT Press and McGraw-Hill. pp. 1024–1027.
[3] Ab. Faisal N.Abu Khazm, “Kernelization Algorithm for Vertex Cover Problem”, PhD thesis, Dept. of Computer Science, University of Tennessee, 2003.
[4] Rajiv Kalapada,”Hybrid Evolutionary Algorithm for Minimum Vertex Cover for Random Graph”, Proceedings of the 9th annual conference on Genetic and evolutionary computation GECCO 07 (2007).
[5] H.Bhasin and Nishant Sharma,”Randomized algorithm approach for solving PCP”,IJCSIT,2012.
[6] D.Kumlander,”Applying AI and Incomplete Solution Principles to Solve NP-hard Problems in the Real-Time Systems”, Proceedings of the 10th WSEAS International Conference on Computers, Vouliagmeni, Athens, Greece, July 13-15, 2006.
[7] Karakostas, George (2004). "A better approximation ratio for the Vertex Cover problem". *ECCC TR04-084*.

AUTHORS PROFILE

Harsh Bhasin has completed his B. Tech in Computer Science and Engineering, M. Tech (C. E.). He is presently working in computergrad.com and a member of International Association of Computer Science and Information Technology. He has published many papers in the domain of Random Number generation and NP hard problem by cellular automata; Cryptography, machine learning, equation solving and NP hard problems by Genetic Algorithm; conversion of south Asian languages by Natural Language Processing. His Various papers have been published in IJCA online, IJCSE, IJCSIT, IJCST and IEEE and National and International conferences.

Geetanjli Ahuja is doing her M. Tech in Computer Science and Engineering from Amity University, India.