# Modified Approach for Solving Maximum Clique Problem

Jaspreet Singh
*M.Tech Scholar, LPU, Phagwara. Punjab, India.*

***ABSTRACT:***
*Maximum Clique problem is an NP Complete Problem which finds its applications in various fields ranging from networking to determination of structure of a protein molecule. The work suggests the solution of above problem with the help of Genetic Algorithms. Clique problem requires finding out all fully connected sub-graphs of a particular graph. This paper investigates the power of genetic algorithms at solving the maximum clique problem. Another feature of algorithm is based on population-driven search where the statistical properties of the initial population are controlled to produce efficient search. The technique can be extended to many unsolvable problems and can be used in many applications from loop determination to circuit solving.*

***KEYWORDS:*** *MAXCLIQUE, Roulette Wheel Selection, Genetic Algorithm, Maximum Clique Problem (MCP).*

## I. INTRODUCTION:

### 1.1 Maximum Clique Problem

Assume that the finite undirected simple graph $G = (V,E))$ is given, where $V$ is the set of nodes, $V =$⊠$N$ , $E$ is the set of edges. The arbitrary full graph is called a clique. The clique, which does not contain other cliques, is called a maximal Clique. The largest maximal clique is called a maximum clique. To extract all maximal cliques from the graph G. Many algorithms have been described to solve this problem. The best solution now a days is a procedure where the complexity is linear to the number of maximal cliques [1,2]. The theory and algorithms described in this paper can solve the problem. We assume that the graph G is presented in the form of an adjacency matrix X of order N*N, the main diagonal of which has zeros. Given an undirected graph G = ( V, E ), a clique S is a subset of V such that for any two elements u, v ε S, ( u, v ) ε E. Using the notation ES to represent the subset of edges which have both endpoints in clique S, the induced graph GS = ( S, ES ) is complete. To Find Maximum clique in a graph is an NP-hard problem, called the maximum clique problem (MCP). Cliques are intimately related to vertex covers and independent sets. Given a graph G, and defining E* to be the complement of E, S is a maximum independent set in the complementary graph G* = ( V, E* ) if and only if S is a maximum clique in G. It follows that V – S is a minimum vertex cover in G*.

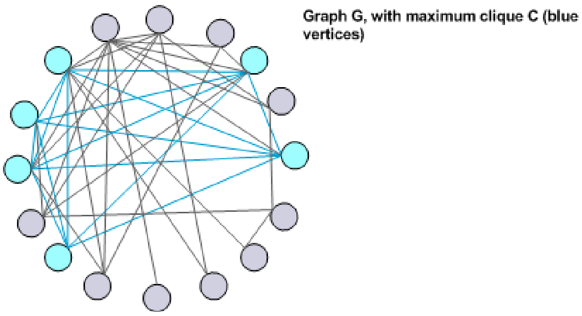

**Fig. 1** An Example of Clique.

In other words a **clique** in an undirected graph $G = (V, E)$ is a subset of the vertex set $C \subseteq V$, such that for every two vertices in $C$, there is an edge connecting the two vertices. This is equivalent to saying that the sub graph induced by $C$ is complete. A maximal clique is a clique that cannot be extended by including one more adjacent vertex to it, means and a clique which does not exist exclusively within the vertex set of a larger clique. A maximum clique is a clique of the largest possible size in a given graph. The clique number $\omega(G)$ of a graph $G$ is defined the number of vertices in a maximum clique in $G$. The intersection number of $G$ is also termed as the smallest number of cliques that together cover all edges of $G$.

The opposite of a clique is observed as an independent set, in the sense that every clique which corresponds to an independent set in the complement graph. The cliques cover problem concerns finding as few cliques as possible that include every vertex in the graph. A related concept is a bi-clique, a complete bipartite sub graph. The bipartite dimension of a graph is the minimum number of bi-cliques needed to cover all the edges of the graph.

## 1.2 GENETIC ALGORITHMS

Genetic algorithms are the closest computation model to natural evolution. Their success at searching complex non-linear spaces and general robustness has led to their use in a number of practical problems such as scheduling, financial modeling and optimization. The inventor of genetic algorithms, John Holland, took his inspiration for them from nature. Genetic algorithms contain a population of individuals, each of which has a known fitness. The population is evolved through successive generations; the individuals in each new generation are bred from the fitter individuals of the previous generation. Unlike Natural Evolution which is continuous indefinitely, we have to decide when to stop our GA. As with the breeding of domestic animals, we choose the individuals to breed from to drive the population's evolution in the direction we want it to go. As with domestic animals, it may take many generations to produce individuals with the required characteristics. Inside a computer an individual's fitness is usually calculated directly from its DNA and so only the DNA need be represented. Usually genetic algorithms represent DNA by a fixed length vector. Where a genetic algorithm is being used for optimization, each individual is a point in the search space and is evaluated by the fitness function to yield a number indicating how good that point is. If any point is good enough, the genetic algorithm stops and the solution is simply this point. If not then a new population, containing the next generation is bred.

The breeding of a new generation is inspired by nature; new vectors are bred from the fitter vectors in the current generation, using either asexual or sexual reproduction. In asexual reproduction, the parent vector is simply copied.Chromosomes are selected from the population to be parents to crossover. The problem is how to select these chromosomes. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzmann selection, tournament selection, rank selection, steady state selection and some others. Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where are placed all chromosomes in the population, every chromosome has its place big accordingly to its fitness function.

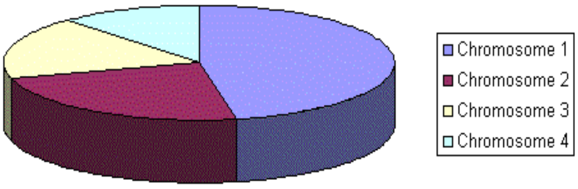

**Fig.2** Roulette Wheel Selection

Then a marble is thrown there and selects the chromosome. Chromosome with bigger fitness will be selected more times. Figure 3 shows a child vector being created by mutating a single gene where each gene is represented by a single bit. With sexual reproduction, two of the fitter vectors are chosen and the new vector is created by sequentially copying sequences alternately from each parent. Typically only two or three sequences are used, and the point(s) where the copying crosses over to the other parent is chosen at random. This is known as crossover. Figure 4 shows a child being formed firstly by copying four genes from the left-hand parent then the three remaining genes are copied from the right-hand parent.

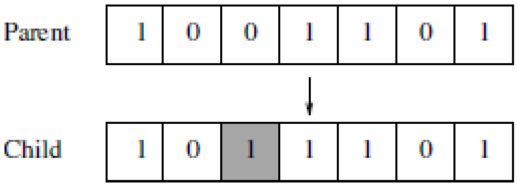

**Fig. 3** Mutation

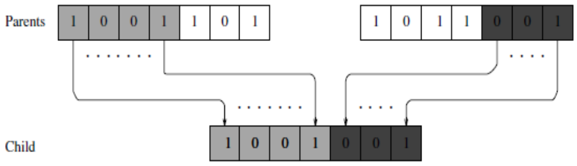

**Fig. 4** Crossover

Holland in his paper "Genetic Algorithms and the Optimal Allocation of Trials" [Hol73] shows, via his schemata theorem, that in certain circumstances genetic algorithms make good use of information from the search so far to guide the choice of new points to search. Figure 5 shows the genetic algorithm cycle. The schemata theorem requires the vector representation and fitness function be designed so that the required solution can be composed of short fragments of vectors which, if present in a vector, give it a relatively high fitness regardless of the contents of the rest of the vector. These are known as building blocks. They can be thought of as collections of genes which work well together.

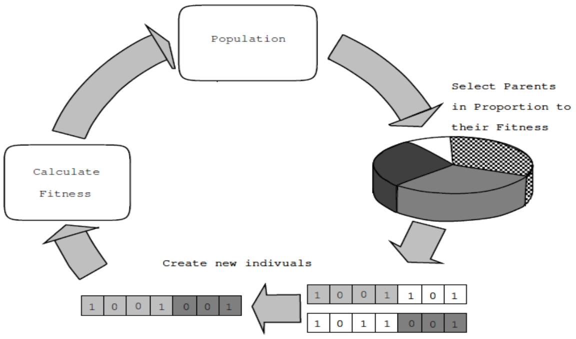

**Fig. 5** The Genetic Algorithm Cycle.

### 1.3 OUTLINE OF THE BASIC GENETIC ALGORITHM

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete

a. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

b. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

c. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).

d. **[Accepting]** Place new offspring in a new population

4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

### 1.4 MAXIMUM CLIQUE PROBLEM IS NP HARD

Examples of difficult problems, which cannot be solved in "traditional" way, are NP problems. There are many tasks for which we know fast (polynomial) algorithms. There are also some problems that are not possible to be solved algorithmically. For some problems was proved that they are not solvable in polynomial time. But there are many important tasks, for which it is very difficult to find a solution, but once we have it, it is easy to check the solution. This fact led to **NP-complete** problems. NP stands for nondeterministic polynomial and it means that it is possible to "guess" the solution (by some nondeterministic algorithm) and then check it, both in polynomial time. If we had a machine that can guess, we would be able to find a solution in some reasonable time. Studying of NP-complete problems is for simplicity restricted to the problems, where the answer can be yes or no. Because there are tasks with complicated outputs, a class of problems called **NP-hard** problems has been introduced. This class is not as limited as class of NP-complete problems. For NP-problems is characteristic that some simple algorithm to find a solution is obvious at a first sight - just trying all possible solutions. But this algorithm is very slow (usually O(2^n)) and even for a bit bigger instances of the problems it is not usable at all. Today nobody knows if some faster exact algorithm exists. Proving or disproving these remains as a big task for new researchers. Today many people think, that such an algorithm does not exist and so they are looking for some alternative methods – example of these methods are genetic algorithms. Examples of the NP problems are Maximum Clique Problem, Travelling Salesman Problem or Knapsack Problem.

## 1.5 APPLICATIONS OF MAXIMUM CLIQUE PROBLEM

The MAXCLIQUE Problem has many real world applications. It is encountered in many different fields in which either the underlying problem can be formulated as the MAXCLIQUE problem or finding the maximum clique is a precondition of solving the problem. Based on those applications, a collection of much diversified test graphs for the MAXCLIQUE problem has been created for evaluating the performance of algorithms for the MAXCLIQUE problem. They are available at ftp://dimacs.rutgers.edu/pub/challenge/graph/ and consist of graphs derived from different problems such as coding theory, fault diagnosis and printed circuit board testing.

### 1.5.1 Coding theory

A common problem in coding theory is to find a binary code as large as possible that can correct a certain number of errors for a given binary word. A binary code is a set of binary vectors. The Hamming distance between two binary vectors is defined as the number of positions in which the two vectors have different values. A maximum clique of H(n,d) represents the maximum number of binary vectors of size n with Hamming distance greater than or equal to d. Therefore, if we find the maximum clique C in H(n,d), any binary code consisting of vectors represented by the vertices in C is able to correct (d-1)/2 errors.

### 1.5.2 Fault diagnosis

Fault diagnosis plays a very important role in studying the reliability of large multiprocessor systems. The goal is to identify all faulty processors (units) in the system. In the model designed by Berman and Pelc [1], the system is represented by an undirected graph G = (V,E) whose vertices are processors and where edges are communication links.

### 1.5.3 Printed circuit board testing

A printed circuit board tester involves placing probes onto a board. A probe can determine if a portion of a board is working correctly. Since probes have a particular size, not every component can be checked in one pass. The problem of maximizing the number of components checked in one pass can be formulated as a clique problem: each node connects a component and an edge represents two nodes that are not too close to be checked simultaneously. A clique in this graph is then a set of components that can be checked in one pass.

## 1.6 OTHER TECHNIQUES OF FINDING MAXIMUM CLIQUE PROBLEMS

### 1.6.1 Simulated Annealing

Simulated annealing is a randomized neighborhood search algorithm inspired by the physical annealing process, where a solid is first heated up in a heat bath until it melts, and then cooled down until it solidifies into a low-energy state. It was first introduced by Kirkpatrick, Gelatt and Vecchi in 1983 [7]. This heuristic technique considers the solutions of a combinatorial optimization problem corresponding to the states of the physical system and the cost of a solution is equivalent to the energy of the state. A simulated annealing algorithm basically works as follows. First, a tentative solution in the state space is generated usually at random. Then the next state is produced from the current one. The new state is evaluated by a cost function f. If it improves, the new state is accepted. If not, the new state is accepted with probability $e^{\Delta f/\tau}$ , where $\Delta f$ is the difference of the cost function between the new state and the current state, and $\tau$ is a parameter usually called the temperature in analogy with physical annealing, which is varied during the optimization process [8]. The simulated annealing heuristic has been ranked among one of the best heuristics for the MAXCLIQUE problem at the 1993 DIMACS challenges [6].

### 1.6.2 Neural Networks

An artificial neural network (ANN) is a parallel system inspired by the densely interconnected, parallel structure of the mammalian brain information- processing system [5]. Some mathematical models of the biology nervous systems show that the temporal evolution is controlled by a quadratic Lyapunov function (also called energy function), which is iteratively minimized as the process evolves. This feature can be applied to many combinatorial optimization problems. More than ten algorithms have been proposed for solving the MAX-CLIQUE problem using neural networks. They encode the problem in different models, but most of them are based on the Hopfield model [5] and its variations. The problem is solved via several discrete (deterministic and stochastic) and continuous energy-descent dynamics. In general, algorithms based on neural networks can find significantly larger cliques than other simpler heuristics but the running time is slightly longer. On the other hand, comparing to those more sophisticated heuristics, they obtained significantly smaller cliques on average but were considerably faster [8].

### 1.6.3 Tabu Search

Tabu search is a modified local search algorithm, in which a prohibition-based strategy is employed to avoid cycles in the search trajectories and to explore new regions in the search space [8]. A tabu list is used to store historical information on the search path to prevent the algorithm from going back to recently visited solutions. Tabu solutions are accepted if they satisfy some aspiration level condition. Several tabu search algorithms for the MAXCLIQUE problem have been developed in the past ten years. They basically have the same structures but change the definition of the search space, the ways that tabu lists are used and the aspiration mechanism. In Battiti and Protasi's algorithm [3], a reactive local search method is used so that the size of the tabu list can be automatically determined. Also, an explicit restart procedure influenced by memory is activated to introduce diversification. The worst case complexity per iteration of this algorithm is O(max(|V| , |E|)) where V is the vertex set and E is the edge set of the graph. The running time of the algorithm is better than those presented at the Second DIMACS Implementation Challenge [6]. There are also many simple heuristics that have been used to solve the MAX-CLIQUE problem such as the sequential greedy heuristics and local search heuristics. They usually have better running times than those advanced heuristics algorithms discussed above, but the quality of the results is worse on the average.

## II. LITRATURE REVIEW

This observation was first mathematically formulated by John Holland in 1975 in his paper, "Adaptation in Natural and Artificial Systems" [5]. Usually the algorithm breeds a predetermined number of generations; each generation is populated with a predetermined number of fixed length binary strings. These binary strings are then translated (decoded) into a format that represents suitable parameters either for some controller, or as an output.The product resulting from evolution (whether natural or simulated) is not simply discovered by a random search through the problem state space, but by a directed search from random positions in that space. In fact, according to Goldberg, the simulated evolution of a solution through genetic algorithms is, in some cases, more efficient and robust than the random search, enumerative or calculus based techniques. The main reasons given by Goldberg are the probability of a multi-modal problem state space in non-linear problems, and that random or enumerative searches are exhaustive if the dimensions of the state space are too great [4].David R. Wood, “An Algorithm for finding maximum clique in a graph”, 1997 Elsevier Science, introduced a branch-and-bound algorithm for the maximum clique problem which applies existing clique finding and vertex coloring heuristics to determine lower and upper bounds for the size of a maximum clique[9]. Patric R. J. Ostergard, “A Fast Algorithm for the maximum clique problem”, 2002 Elsevier Science, given a branch-and-bound algorithm for the maximum clique problem—which is computationally equivalent to the maximum independent (stable) set problem—is presented with the vertex order taken from a coloring of the vertices and with a new pruning strategy. The algorithm performs successfully for many instances when applied to random graphs and DIMACS benchmark graphs [10].Xinshun Xu, Jun Ma, Jingsheng Lei, “Ant Colony Optimization for the Maximum Clique Problem” IEEE-ICNC 2007 introduced an evolutionary approach in which main task is to search for maximum cost path in a graph. Artificial Ants walk through graph and looking for high quality paths. Better results are found as emergent result of global cooperation among ants in colony.

Li Lu, Yunhong Gu, Robert Grossman, “dMaximalCliques: A Distributed Algorithm for Enumerating All Maximal Cliques and Maximal Clique Distribution” IEEE- ICDMW-2010 presents a distributed algorithm which can obtain clique information from million-node graphs. It has used distribution of size of maximal cliques in a graph as a new measure for measuring structural properties of a graph.

R.Rama, Suresh Badarla and Kamala krithivasan, “Clique-detection algorithm using clique-self-assembly”, IEEE BIC-TA.2011, proposed a brute force algorithm where a large graph is being decomposed and these decomposed parts are cliques [11].

Hakan Yıldız, Christopher Kruegel, “Detecting Social Cliques for Automated Privacy Control in Online Social Networks”, Fourth International Workshop on Security and Social Networking, Lugano (19 March 2012) proposed a privacy control approach that addresses this problem by automatically detecting social cliques among the friends of a user. To find cliques, given a small number of friends (seed), uses the structure of the social graph to generate an approximate clique that contains this seed. The cliques found by the algorithm can be transformed directly into friend lists, making sure that a piece of sensitive data is exposed only to the members of a particular clique.Harsh Bhasin, Rohan Mahajan, “Genetic Algorithms Based Solution To

Maximum Clique Problem", ISSN: 0975-3397 Vol. 4 No. 08 Aug 2013, suggests the solution of above problem with the help of Genetic Algorithms (GAs). The work also takes into consideration, the various attempts that have been made to solve this problem and other such problems [12].

## III. PRESENT WORK

### 3.1 Significance

Since the MAXCLIQUE problem has important applications, designing an algorithm with good performance becomes necessary and important. A lot of algorithms for solving the MAXCLIQUE problem have been proposed in the literature since the 1990's. The early work focused on the exact algorithms. Some of them achieved success on small graphs (less than 400 vertices). For example, Balas and Yu [2] improved the implicit enumeration algorithm by reducing the number of sub-problems generated from each node of the search tree, which in turn, reduced the size of the whole search tree. But still, the running time of exact algorithms increases exponentially with the size of graphs. So it is not practical to solve large problems by exact algorithms. The next step is to approximate the maximum clique size to be within a certain factor of the optimal solution. Therefore, it is necessary to solve the problem using heuristic algorithms. Heuristic algorithms cannot guarantee the quality of solutions but in practice they have been observed to produce very good solutions.

**My area of research concentrates on solution of Maximum Clique Problem using a genetic algorithm which is a parallel search procedure inspired by the mechanisms of evolution in natural systems and the scenarios under which the solutions are applicable, the various qualitative parameters satisfied by the solutions and their associated costs in order to meet the current and future resource pool of a dynamic requirements in the most efficient way based on various qualitative and quantifiable parameters, so that virtualization or multi-tenancy can be easily deployed for providing various search services.**

### 3.2 Objective

[1] To identify various qualitative and quantifiable parameters for the fitness function to make intelligent decisions regarding fitness evaluation of chromosomes in a given population.

[2] To design a solution for finding Maximum Clique Problem based upon extensive research of journals and articles in the field.

[3] To identify various Clique finding algorithms and their solutions for helping Genetic Algorithm based solution to Maximum Clique Problem.

[4] Implementation of the Maximum Clique Finding Algorithm in MATLAB.

### 3.3 Methodology

In order to build the Genetic Algorithm based solution to Maximum Clique Problem the following methodology is followed:-

[1] **Identification**
In this step the requirement analysis is done. This requires a task analysis to be done to determine the requirements, the inputs and outputs the prospective users.

[2] **Conceptualization**
In this the proposed program is designed to understand and define the specific relationships and interactions in the problem domain. The key concepts, the relationships, processes and control mechanisms are determined. This is the initial stage of knowledge acquisition.

[3] **Formalization**
This involves organizing the key concepts and information into formal representations i.e. rules for the Encoding and Crossover. It involves deciding the attributes to be determined to solve the problem and to build the initial mutated result.

[4] **Implementation**
This involves mapping of the formalized population into a framework of the development tool (MATLAB) to build a working Matrix. The contents of matrix structures, inference rules and control strategies established in the previous stages are organized into suitable format.

### 3.4 Sources of Data

The first population for the matrix computation will be developed through extensive study of journals, books, white papers etc. as well as experts in the field of Advanced Data Structures, Heuristic Searching Techniques and Evolutionary Search Strategies etc.

## REFERENCES:-

[1] P. Berman and A. Pelc, “Distributed Fault Diagnosis for Multiprocessor Systems” Proc. of the 20th Annual Int. Symp. On Fault-Tolerant Computing (Newcastle, UK), 1990, pp. 340-346.

[2] E. Balas and C. S. Yu, “Finding a Maximum Clique in an Arbitrary Graph" SIAM J. Comput., 14, 1986, pp. 1054-1068.

[3] R. Battiti and M. Protasi, “Reactive Local Search for the Maximum Clique Problem” Technical Report TR-95-052, International Computer Science Institute, Berkeley, CA, 1995.

[4] D. E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning” Addison-Wesley, Boston, MA, 1989.

[5] J.Hertz, A. Krogh and R. G. Palmer, “Introduction to the Theory of Neural Computation” Assison-Wesley, Redwood City, CA, 1991.

[6] D. S. Johnson and M. A. Trick (eds.), “Cliques Coloring and Satisfiability, Second DIMACS Implementation Challenge” DIMACS 26, American Mathematical Society, 1996 (see also http://dimacs.rutgers.edu/Volumes/Vol26.html).

[7] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, “Optimization by Simulated Annealing” Science, 220, 1983, pp. 671-680.

[8] P. M. Pardalos and J. Xue. “The Maximum Clique Problem” Journal of Global Optimization, 4, 1994, pp. 301-328.

[9] David R. Wood, “An algorithm for Finding a maximum clique in a graph” 1997 Elsevier Science B.V. PII S0167-6377(97)00054-0

[10] Patric R. J. Ostergard, “A Fast Algorithm for the maximum clique problem”, 2002 Elsevier Science B.V. PII: S0166-218X (01)00290-6

[11] R.Rama, Suresh Badarla and Kamala krithivasan, “Clique-detection algorithm using clique-self-assembly”, IEEE DOI 10.1109/BIC-TA.2011.32

[12] Harsh Bhasin, Rohan Mahajan, “Genetic Algorithms Based Solution To Maximum Clique Problem”, ISSN: 0975-3397 Vol. 4 No. 08 Aug 2013