MAXCLIQUE Problem Solved Using SQL

Jose Torres-Jimenez,
Nelson Rangel-Valdez,
Loreto Gonzalez-Hernandez
Information Technology Laboratory
CINVESTAV-Tamaulipas, Victoria Tamps., MEXICO
Email: jtj@cinvestav.mx,
nrangel@tamps.cinvestav.mx,
agonzalez@tamps.cinvestav.mx

Himer Avila-George
Departamento de Sistemas Informáticos y Computación (DSIC)
Universidad Politécnica de Valencia
Valencia, Spain
Email: hiavgeo@posgrado.upv.es

***Abstract*—This paper aims to show that SQL queries can be used to solve a well-known combinatorial optimization problem, the Maximum Clique Problem (MAXCLIQUE). This problem arises in many real world applications as computer vision and pattern recognition or coding theory to mention some of them. A clique of a graph is a complete subgraph, i.e., a graph where every pair of vertices is an edge. The MAXCLIQUE problem searches for the clique of the largest cardinality in a graph (the one with the greatest number of nodes). We propose a model based on SQL queries to solve this problem. We test our models in 62 random instances. Results show that the use of simple queries can yield solutions for the MAXCLIQUE problem in an easy and accessible form.**

Keywords- Optimization; MAXCLIQUE; SQL query.

I. INTRODUCTION

Combinatorial optimization is a branch of applied mathematics and computer science that is used to solve problems like circuit design [1], scheduling [2], software testing [3], [4] and many other problems related with real world applications. A lot of problems presented in combinatorial optimization are best understood when they are abstracted in mathematical structures like graphs. Graph theory is the field of mathematics and computer sciences that studies all the aspects related with graphs. The importance of studying the combinatorial optimization problems is the wide application it has in real world problems.

One of the basic problems in graph theory is the MAXIMUM CLIQUE problem (MAXCLIQUE). This problem can be defined as the search of a complete subgraph of maximum cardinality, i.e. it contains the maximum number of vertices. Applications of this problem arises in coding theory [5], computer vision and pattern recognition [6], fault diagnosis [7] and protein structure similarity [8].

Several methods have been used to solve the MAXCLIQUE problem. In the literature can be found exact and non-exact approaches. A survey about this problem can be found in [9].

Most of the techniques used to solve the MAXCLIQUE problem rely in the use of a high level procedural language like C [10], [11]. In this paper we propose two models to solve this problem using a well known non-procedural data access sublanguage, the Structured Query Language (SQL) [12]. SQL is easy to use and allows the users to express the desired results of a query in a high-level data sublanguage.

To the best of our knowledge, there is no reported approach that uses SQL queries to solve MAXCLIQUE instances. Moreover, we found no approach that solves any combinatorial optimization problem using SQL queries. The simplicity of the SQL language and the availability of database manager systems that allow the use of SQL motivate us to design a new approach that, based on SQL queries, could solve instances of the MAXCLIQUE problem. Basically, we present a query model which can be used to determine if a given graph has a clique of size k or smaller. The model was tested in a set of random generated instances.

The rest of the paper is organized as follows: section 2 presents the notation and formal definition of the MAXCLIQUE problem. Section 3 describes the optimization model based on SQL that solves this problem. Section 4 shows the experimental design used to test the proposed solution. Section 5 presents the conclusions derived from the results obtained when solving MAXCLIQUE instances through SQL queries.

II. MAXIMUM CLIQUE PROBLEM

In this section we give a formal definition for the MAXCLIQUE problem. After that, we show the solution of an instance of the MAXCLIQUE problem. Finally, we end the section analyzing the search space of the problem.

A. Formal Definition of the Maximum Clique Problem (MAXCLIQUE)

A graph $G = (V, E)$ is described by a set V of nodes and a set E of edges (or links between pair of nodes). The number of nodes of the set V represents the order of the graph G (denoted by $|G|$). A graph G is called *clique* when every pair of nodes $i, j \in V$, where $i \neq j$, is an edge

 ISBN:978-1-61208-115-1

$(i, j) \in E$. The size of a clique is its order. From now on, a clique will be denoted by $\mathcal{K}$ and its size by $|\mathcal{K}|$.

The MAXCLIQUE problem can be defined as follows: given a graph G, which subgraph $\mathcal{K}^* \subseteq G$ is the clique with the maximum possible cardinality $|\mathcal{K}^*|$? The set of subgraphs of G that answers this question is defined by the Equation 1. The subgraph $\mathcal{K}^*$ is any of the subgraphs of G found in $\mathcal{K}^*_{all}$.

$$\mathcal{K}^*_{all} = \{\mathcal{K} : \mathcal{K} \subseteq G, |\mathcal{K}| = \max\{|\mathcal{K}| : \mathcal{K} \subseteq G\}\} \quad (1)$$

B. Example of a MAXCLIQUE instance

Following the definition already given, an instance of the MAXCLIQUE problem is a graph G. Figure 1 shows an instance of the MAXCLIQUE problem. This instance has 5 nodes and 5 edges.

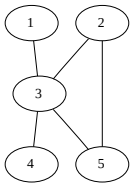

Figure 1. Instance of the MAXCLIQUE problem. The graph $G = (V, E)$ has the set $V = \{1, 2, 3, 4, 5\}$ and the set $E = \{(1, 3), (2, 3), (2, 5), (3, 4), (3, 5)\}$.

Figure 2 shows the SQL instance of the graph $G = (V, E)$ shown in Figure 1. The the set of nodes V is represented in the table $nodes$; the field v is used to label the nodes of the graph and the field s to assign a weight to the nodes. The set of edges E is represented in the table $edges$; this table has two fields used to represent each edge of the graph as pair of nodes. In order to find clique graphs of size smaller than the predefined value k, a virtual node 0 and virtual edges between it and the original nodes are added to the graph instance; this will result in a participation of the node 0 in every clique found, but this node will not count in the size of the clique because its value in the field s is zero.

A SQL query can be used to solve the instance of MAXCLIQUE shown in Figure 2. First of all, this query involves a subset of size k. The query will try to find the greatest subset of size equal to or smaller than k that be a clique; it is possible because when there is no subset of size k the query looks by a subset of size $k - 1$ by including one time the zero node in the solution. Moreover, smaller subsets are possible by including as much zero nodes to the solutions as it is necessary.

The SQL query is shown in Table I. The *FROM* clause generates all the possible combinations of five copies of table $nodes$. The *WHERE* clause filters the combinations allowing only those in which a clique exists. The *ORDER BY* clause is used because the results of the query will give all the possible combinations of nodes that yields a clique.

$edges$	
n_1	n_2
0	0
0	1
0	2
0	3
0	4
0	5
1	3
2	3
2	5
3	4
3	5

(a)

$nodes$	
v	s
0	0
1	1
2	1
3	1
4	1
5	1

(b)

Figure 2. SQL tables encoding the instance of the MAXCLIQUE problem shown in Figure 1.

The evaluation of the existence of a clique is made by the clauses *EXISTS* which ask for the existence of all the edges required to form a clique. The descending order will give the maximum clique at the first entry of the results of the SQL query and the keywords *LIMIT* 1 specify that we only want the first result (in case there is more than one solution). The name of the nodes that forms the clique and the size of the clique is specified in the *SELECT* clause.

Table I
SQL QUERY THAT SOLVES THE MAXCLIQUE PROBLEM INSTANCE SHOWN IN FIGURE 2.

SELECT
$V_1.v$, $V_2.v$, $V_3.v$, $V_4.v$,$V_5.v$,
$(V_1.s + V_2.s + V_3.s + V_4.s + V_5.s)$
FROM
$nodes$ as V_1, $nodes$ as V_2, $nodes$ as V_3, $nodes$ as V_4, $nodes$ as V_5
WHERE
$EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_1.v$ AND $E.n_2 = V_2.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_1.v$ AND $E.n_2 = V_3.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_1.v$ AND $E.n_2 = V_4.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_1.v$ AND $E.n_2 = V_5.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_2.v$ AND $E.n_2 = V_3.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_2.v$ AND $E.n_2 = V_4.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_2.v$ AND $E.n_2 = V_5.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_3.v$ AND $E.n_2 = V_4.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_3.v$ AND $E.n_2 = V_5.v$)
AND $EXISTS$(SELECT * FROM $edges$ as E WHERE ($E.n_1 = V_4.v$ AND $E.n_2 = V_5.v$)
ORDER BY 6 **DESC LIMIT** 1

The solution of the SQL query presented in Table I is shown in the Table II. The maximum clique size is 3. The columns represent the combination of the maximum clique and its size. Given that we search for a clique of size k, the number of columns in the solution will be $k + 1$, the first k columns are nodes that form the clique (a zero indicates a virtual node and must be ignored), the size of the clique is given in the last column.

Table II
SOLUTION OF THE MAXCLIQUE INSTANCE SHOWN IN FIGURE 2.

Clique					
$V_1.v$	$V_2.v$	$V_3.v$	$V_4.v$	$V_5.v$	*size*
0	0	2	3	5	3

 ISBN:978-1-61208-115-1

C. Complexity Analysis of the MAXCLIQUE Problem

For a MAXCLIQUE instance, the search space in the SQL query is defined by the cartesian product performed in the clause *FROM*. Given that this product is between k tables of size $N+1$ (the size of the clique and the number of nodes plus 1, respectively), the set of possible solutions for the problem has a cardinality of $(N+1)^k$.

Several exact approaches have been proposed in the literature that solve the MAXCLIQUE problem [9]. These approaches generally work in the search space defined by all the subsets of size k or less of the set of nodes V of a graph $G=(V,E)$. This search space is equivalent to the cardinality of the power set of the set of nodes V, denoted by $|\mathcal{P}(V)|$ and defined in Equation 2.

$$|\mathcal{P}(V)| = \sum_{i=0}^{N} \binom{N}{i} \tag{2}$$

A comparison between the search space that potentially is searched by the SQL query is greater than the real search space. We present experimental evidence that the solution for smaller instances of the MAXCLIQUE problem are worth of attention through a SQL approach, taking into account that the high level of SQL avoids the programming of complex routines in low level languages.

Next section presents the generalization of the solution for the particular instance of the MAXCLIQUE problem already presented.

III. SQL APPROACH FOR SOLVING THE MAXCLIQUE PROBLEM

Given an instance of the MAXCLIQUE Problem as a graph $G=(V,E)$ and an integer k, $1 \leq k \leq N$, we propose an exact approach that finds a clique of size k or smaller in G. The approach consists on generating a query in the Standard Query Language (SQL). Once that the query has been created, it is executed in a database management system so that the solution is obtained.

The query shown in Table I presents the SQL query required to solve the particular MAXCLIQUE instance shown in Figure 2. Table III shows the generalization of that query such that any MAXCLIQUE instance defined according with the definition given at Section II-A can be solved using a SQL query and a database management. The query is given using the BNF notation [13]. The query uses the tables $edges$ and $nodes$. The table $edges$ contains the nodes of the graph V been solved; an extra virtual node called 0 is added to this set. A column associates the value 1 with each node in the original set V and the value 0 with the node 0. The table $edges$ is the set of edges of the instance; this set includes extra virtual edges between node 0 and the rest of the nodes. The existence of edges between node 0 and the rest of the nodes allows this node to participate in any existing clique, but its contribution to the size of the clique is 0 so that cliques formed by nodes of the original graph are preferred. In general, the inclusion of the node 0 and the edges with this node will enable the query to give as answer cliques of size smaller than k when a clique of size k does not exist. The solution of this query reports the nodes in the maximum clique found and the size of such clique.

Table III
BNF FORMAT OF THE GENERAL SQL QUERY THAT SOLVES THE MAXCLIQUE PROBLEM.

SELECT
  <subset of nodes>,
  <size of clique>
FROM
  <subset definition>
WHERE
  <constraint definition>
ORDER BY $< k+1 >$ **DESC LIMIT** 1

<subset of nodes> ::=
  $V_1.v, \ldots, V_K.v$
<size of clique> ::=
  $(V_1.s + V_2.s + \ldots + V_K.s)$
<subset definition> ::=
  $nodes$ as V_1, $nodes$ as V_2, ..., $nodes$ as V_K
<constraint definition> ::=
  $EXISTS(SELECT * FROM\ edges\ WHERE$
    $edges.n1 = V_1.v\ AND\ edges.n2 = V_2.v)$
    $AND \ldots$

The structure of the query shown in Table III can be described as follows: given the graph $G=(V,E)$ as the SQL tables $edges$, $nodes$ with fields n_1 and n_2 in $edges$ and v, s in $nodes$, the *FROM* clause indicates the cartesian product of k tables as the nodes in the maximum clique of the instance that is going to be solved. The *SELECT* clause indicates the subset of maximum size that is a clique in the instance G. In addition, the *SELECT* clause include an extra field that represents the size of the clique found in the solution process. The *WHERE* clause will contain the condition that must be met so that a clique is formed by the SQL query; these instructions are logical ANDs of query's asking for the existence of every possible edge that must be contained in the clique, i.e., the query will ask for the existence of $\binom{k}{2}$ edges in this clause. Finally, the SQL query will sort the results by the extra field representing the size of the cliques. The descending order in combination with the clause *LIMIT 1* allow to identify a solution returned by the query; the descending order on the size of the cliques makes that the largest cliques appear as the first rows in the results, the clause *LIMIT 1* extracts only the first of them. Note that a convenient use of the clause *LIMIT 1* allows an extension in the results reported by the query, i.e. the query can report the c largest cliques found in the instance by specifying it in this clause (something like *LIMIT c*). In this way, if different cliques of the same size exists, they will be reported in the following tuples of the results.

In the next section is presented an experimental design to solve random instances of the MAXCLIQUE problem.

 ISBN:978-1-61208-115-1

IV. Experimental Design

In this section we present the experimental design done to test the model based on a SQL query to solve the MAXCLIQUE problem. In order to test the performance of the query we solved 62 random instances of the MAXCLIQUE problem. The generators of the instances are described in the next subsection.

A. Random Instance Generators

Two models were considered for the generation of random instances of the MAXCLIQUE problem. The first model, or model A, is the well known Erdös-Rényi model [14]. This model works as follow: given the number of nodes n and a probability p, each edge $(i, j) \in E$ of the resulting graph $G = \{V, E\}$ is selected with a probability p, i.e., a random number is generated for each of the $\binom{n}{2}$ possible edges that a graph can have, and those edges with a random number smaller than p will belong to the graph.

Algorithm IV.1 shows the pseudocode of the model A for generation of random graphs. This algorithm takes as input the number of nodes n and the density p and gives as output a graph $G = (V, E)$, where $|V| = n$ and $|E| \approx p \cdot \binom{n}{2}$. Each edge $(i, j) \in E$ is selected with a probability p.

Algorithm IV.1: ModelA(n, p)

comment: O$utput$: $Graph\ G$

for $i \leftarrow 1$ **to** n
do { **for** $j \leftarrow 1$ **to** n
do { $q \leftarrow$ RandomNumber$(0, 1)$
if $q \leq p$
then {Add(G, i, j)
return (G)

A MAXCLIQUE instance using the algorithm previously described is shown in Figure 3. This instance has an number of nodes 6 and a value of p equal to 0.4.

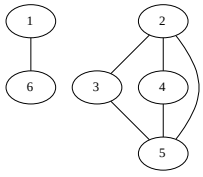

Figure 3. Random Instance of the MAXCLIQUE problem with 6 nodes and $p = 0.4$. The set of nodes is $V = \{1, 2, 3, 4, 5, 6\}$ and the set of edges is $E = \{(1, 6), (2, 3), (2, 4), (2, 5), (3, 5), (4, 5)\}$. A clique of size 3 is formed with the nodes $2, 3$ and 5.

A disadvantage of using only the Erdös-Rényi model is that it doesn't model well the systems where the MAXCLIQUE problem finds its applications. While the Erdös-Rényi model is characterized by small clustering of the nodes with small average length paths, most of the systems that are found in the nature are best represented by highly clustered nodes with small average length paths (or as they are commonly refered, by small-world graphs). Due to this fact, we propose a second set of instances of the MAXCLIQUE problem to test the SQL query. The second model (or model B) is one of most widely used models for the generation of random small-world graphs, the Watts-Strogatz model [15].

The model B starts with a regular lattice and progressively rewires the edges with a probability p. The rewiring process means that every edge $(i, j) \in E$ is disconnected with a probability p (the rewiring probability) from one of its end points and reconnected to another one. A regular lattice is a graph where each node has an edge with its z nearest neighbors, where z is called the coordination number. The pseudocode for the model B is shown in the Algorithm IV.2; this algorithm takes as input the number of nodes n, the coordination number z and the rewiring probability p and returns a random graph based on that values.

Algorithm IV.2: ModelB(n, z, p)

comment: O$utput$: $Graph\ G = (V, E)$

$G \leftarrow$ BuildRegularLattice(n, z)
for $i \leftarrow 1$ **to** n
do { **for** $j \leftarrow 1$ **to** n
do { **if** $(i, j) \in E$
then
{ $q \leftarrow$ RandomNumber$(0, 1)$
if $q \leq p$
then Rewire(G, i, j)
return (G)

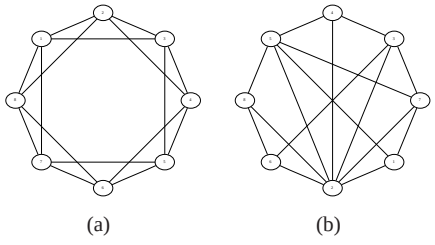

Figure 4. Random graph generated by model B: a) initial regular lattice with 8 nodes and coordination number $z = 2$; b) random graph generated after rewiring the edges with a probability $p = 0.50$.

Figure 4 shows an instance generated by the model B. Figure 4(a) presents the initial regular lattice of 8 nodes with coordination number $z = 2$. Figure 4(b) shows the resulting

graph after applying the generation model B with a rewiring probability of $p = 0.50$.

The next section presents the experimental results obtained from solving the random instances generated by the two models presented in this section. The instances were solved using, the SQL query described in Section III.

B. Computational Experiments

In this section we present the general overview of the experimentation, the instances solved by the SQL query and the features of the hardware and the database management system used in our experimentation. The results are summarized in several tables that contain information about the size of the clique found and the time spent by our approach to find it.

1) Features of the hardware and software: The random instances generator was implemented in C language and compiled with gcc. The instances were generated in a Desktop Computer with an Intel(R) Core(TM) 2 CPU 2400 6400 @ 2.13 Ghz processor, 3Gb of RAM and Ubuntu 8.10 Intrepid Ibex Operating System, and the query resulting from each instance was executed in MySQL 5.0.67.

2) Instances: The experiment was carried out using 62 instances which were created by the random instance generators described in the Section IV-A. Two sets of instances are included in the experiment. The first set (or set S_1) is shown in Table IV; this set is of small instances created using the model A and includes 14 instances with less than 30 nodes. The number of nodes and density of each instance is shown in columns 2 and 3, respectively.

The second set of instances S_2 includes larger instances created using the model B. Table V lists in columns 1 and 4 the different values for n, z, p used to create each of the 48 instances of the set. Basically, the number of nodes considered were $n = \{50, 70\}$, the coordination value z was varied from 5 to 15 and the rewiring probability values were $p = \{0.20, 0.30, 0.40, 0.50\}$.

3) Parameters of the test cases: The parameter required for the experiments was the size of the maximum clique (or k). The experiments were done in 2 different stages. In the first step the set S_1 was considered and tested with $k = 15$, the results from solving these cases are shown in the Table IV. The column 1 shows the instance. The column 2 shows the set of nodes that form the clique (a zero value represents a virtual node, for cliques of size smaller than k). The column 3 contains the size of the maximum clique. The column 4 shows the time consumed by the query.

According with the results shown in Table IV, the SQL query solved almost all the instances with less than 30 nodes and found cliques of size in the range from 3 to 15.

In the second step of the experimentation the set S_2 was considered. Also, in this experiment the value of the clique size was set to $k = 15$. The results from this experiment are shown in Table V. The first three columns shows the results for the instances with $n = 50$ nodes. The last three columns shows the results for the instances with $n = 70$ nodes. For each instance is listed de size of the maximum clique found $|\mathcal{K}^*|$ and the time in seconds spent by the query to find it.

Table IV
RESULTS FROM SOLVING THE SET S_1 OF RANDOM MAXCLIQUE INSTANCES. THE CLIQUE SIZE WAS SET TO $k = 15$.

Cases	n	p	$\mathcal{K}^*$	ω	Time (sec.)
1	10	0.30	0 0 0 0 0 0 0 0 0 0 0 0 3 7 9	3	0.10
2	10	0.45	0 0 0 0 0 0 0 0 0 0 0 0 4 6 7	3	0.15
3	10	0.60	0 0 0 0 0 0 0 0 0 0 0 3 4 6 7 9	5	0.30
4	10	0.75	0 0 0 0 0 0 0 0 0 0 3 4 5 6 7 8	6	0.63
5	10	0.90	0 0 0 0 0 0 0 0 0 1 3 4 6 7 8 10	7	1.68
6	20	0.30	0 0 0 0 0 0 0 0 0 0 0 0 0 8 10 14 18	4	1.40
7	20	0.45	0 0 0 0 0 0 0 0 0 0 0 0 4 6 11 15 20	5	2.99
8	20	0.60	0 0 0 0 0 0 0 0 0 2 5 9 12 14 18 19	7	9.86
9	20	0.75	0 0 0 0 0 0 5 7 9 12 14 16 17 18 19	9	68.96
10	20	0.90	1 2 3 4 6 7 8 10 13 14 15 16 18 19 20	15	1222.83
11	30	0.30	0 0 0 0 0 0 0 0 0 0 0 1 12 14 22 25	5	6.28
12	30	0.45	0 0 0 0 0 0 0 0 0 1 12 14 20 22 25 28	7	19.10
13	30	0.60	0 0 0 0 0 0 0 1 6 8 14 17 22 23 25 27	9	86.22
14	30	0.75	0 0 0 2 5 8 14 15 17 20 22 23 27 28 30	12	815.30

Table V
RESULTS FROM SOLVING THE SET OF LARGE RANDOM MAXCLIQUE INSTANCES WITH THE SQL QUERY. THE VALUE OF THE MAXIMUM CLIQUE SIZE TO BE SEARCHED WAS SET TO $k = 5$.

Instance (n, z, p)	$\lvert\mathcal{K}^*\rvert$	Time (sec.)	Instance (n, z, p)	$\lvert\mathcal{K}^*\rvert$	Time (sec.)
$(50, 5, 0.20)$	6	7.80	$(70, 10, 0.20)$	8	335.51
$(50, 5, 0.30)$	6	6.24	$(70, 10, 0.30)$	9	209.38
$(50, 5, 0.40)$	4	5.36	$(70, 10, 0.40)$	6	151.12
$(50, 5, 0.50)$	4	5.04	$(70, 10, 0.50)$	6	126.88
$(50, 6, 0.20)$	6	10.73	$(70, 11, 0.20)$	9	504.95
$(50, 6, 0.30)$	5	8.68	$(70, 11, 0.30)$	7	235.76
$(50, 6, 0.40)$	5	8.10	$(70, 11, 0.40)$	7	165.93
$(50, 6, 0.50)$	5	8.30	$(70, 11, 0.50)$	7	168.17
$(50, 7, 0.20)$	7	18.72	$(70, 12, 0.20)$	9	637.30
$(50, 7, 0.30)$	7	14.56	$(70, 12, 0.30)$	9	377.82
$(50, 7, 0.40)$	6	14.02	$(70, 12, 0.40)$	7	280.25
$(50, 7, 0.50)$	5	12.20	$(70, 12, 0.50)$	7	229.70
$(50, 8, 0.20)$	8	31.33	$(70, 13, 0.20)$	10	819.89
$(50, 8, 0.30)$	7	18.99	$(70, 13, 0.30)$	9	569.04
$(50, 8, 0.40)$	6	16.90	$(70, 13, 0.40)$	8	428.36
$(50, 8, 0.50)$	5	15.30	$(70, 13, 0.50)$	7	331.47
$(50, 9, 0.20)$	8	35.06	$(70, 14, 0.20)$	10	1423.51
$(50, 9, 0.30)$	7	25.50	$(70, 14, 0.30)$	8	737.20
$(50, 9, 0.40)$	6	23.87	$(70, 14, 0.40)$	8	550.05
$(50, 9, 0.50)$	6	21.83	$(70, 14, 0.50)$	7	458.02
$(50, 10, 0.20)$	9	62.15	$(70, 15, 0.20)$	11	2081.27
$(50, 10, 0.30)$	7	38.39	$(70, 15, 0.30)$	8	719.54
$(50, 10, 0.40)$	7	38.07	$(70, 15, 0.40)$	8	617.33
$(50, 10, 0.50)$	6	32.21	$(70, 15, 0.50)$	7	512.06

The results shown in Table V show instances with an average small clique size (model A generated instances with greater cliques in graph with less nodes). The performance of the SQL query over the set S_2 is better than in the set S_1, i.e. in some instances from the set S_1 the SQL query spent more time to find a clique of almost the same size than in the larger instances found in the set S_2. A natural explanation of this behavior is that small-world graphs tend to be sparse, which weaken the possibility of finding large cliques, and the nodes are highly clustered, which affects the number of different subgraphs that could be a clique; these two characteristic can improves the performance of the SQL

 ISBN:978-1-61208-115-1

query in the sense that the cartesian product exclude a large number of solution during the solution of the instance.

In general, the time spent by the SQL approach to solve the random instances varies from a few seconds to almost two hours. Note that this amount of time consumed by the query to find the maximum cliques is relatively small in comparison with the theoretical search space. For example, the instance 10 shown in Table IV have a clique of size $k = 15$ and the query only spent 1222.83 seconds to find it among the 31^{15} possible solutions. This performance can be mainly explained by the fact that the cartesian product is not done at once, instead it starts with two tables and continue adding them one by one until it is completed. Each time that two tables are combined, those tuples that do not match the conditions specified are left out from the rest of the operation; this action results in a considerable reduction in the search space that contributes to a quick localization of the wished result.

Finally, according with the results showed in this section, we can conclude that the model based on SQL queries can solve instances of the MAXCLIQUE problem when the number of nodes and/or the size of the clique searched are not too large.

V. Conclusions

This paper presents a novel approach for solving the MAXCLIQUE problem using a SQL query. The query was tested in a set of 62 random MAXCLIQUE instances created through the Erdös-Rényi and Watts-Strogatz models. The query performs well in small sparse instances, or instances where the maximum clique is small. The limitations of the model are given by the database management system used to solve the query.

The simplicity of the SQL approach makes it easier to use than procedural languages approaches, in the sense that it does not require complex structures nor programming skills to solve the problem. The performance of the SQL approach depends on the query optimization tools implemented in the database managements system. The results shown that it is possible to solve an important optimization problem using a high level non-procedural languages without coding a line.

Currently we are trying to extend the range in which a SQL query approach for the MAXCLIQUE problem works in reasonable time.

ACKNOWLEDGEMENTS

This research was partially funded by the following projects: CONACyT 58554-Cálculo de Covering Arrays, 51623-Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas.

References

[1] S. N. Bhatt and F. T. Leighton, "A framework for solving VLSI graph layout problems," *J. Comput. Sytem Sci.*, vol. 28, no. 2, pp. 300 – 343, 1984.

[2] Y. N. Sotskov, V. S. Tanaev, and F. Werner, "Scheduling problems and mixed graph colorings," *Optimization*, vol. 51, no. 3, pp. 597–624, 2002.

[3] D. M. Cohen, S. R. Dalal, J. Parelius, and G. C. Patton, "The combinatorial design approach to automatic test generation," *IEEE Softw.*, vol. 13, no. 5, pp. 83–88, 1996.

[4] C. C. Michael, G. E. McGraw, M. A. Schatz, and C. C. Walton, "Genetic algorithms for dynamic test data generation," in *Automated Software Engineering, 1997. Proceedings., 12th IEEE International Conference*. Washington, DC, USA: IEEE Computer Society, 1997, pp. 307–308.

[5] V. Ustimenko and T. Shaska, "On some applications of graphs to cryptography and turbocoding," *Albanian J. Math.*, vol. 2, no. 3, pp. 249–255, 2008.

[6] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Graph matching applications in pattern recognition and image processing," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 2003, pp. II–21–4 vol.3.

[7] P. Berman and A. Pelc, "Distributed probabilistic fault diagnosis for multiprocessor systems," in *Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium*, 1990, pp. 340 –346.

[8] N. Malod-Dognin, R. Andonov, and N. Yanev, "Solving maximum clique problem for protein structure similarity," 2009. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:0901.4833

[9] I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo, "The Maximum Clique Problem," in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. M. Pardalos, Eds. Kluwer Academic Publishers, 1999, vol. A, pp. 1–74.

[10] G. Mulligan and D. G. Corneil, "Corrections to bierstone's algorithm for generating cliques," *J. ACM*, vol. 19, no. 2, pp. 244–247, 1972.

[11] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," *SIAM J. Comput.*, vol. 6, no. 3, pp. 505–517, 1977.

[12] C. Ordonez, "Programming the K-means clustering algorithm in SQL," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2004, pp. 823–828.

[13] J. Friedman, "A computer system for transformational grammar," *Commun. ACM*, vol. 12, no. 6, pp. 341–348, 1969.

[14] P. Erdös and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[15] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

 ISBN:978-1-61208-115-1