The Clique Problem

A Polynomial Time and Non-Heuristic Solution (P=NP)

By

© John Archie Gillis

September 28th, 2020

Gillis Intellectual Property Developments

johnarchiegillis@gmail.com

NOTE:

This is simply another draft version of specific elements of a paper that I am working on. I hope to gather all the best elements for the creation of a LaTex Document for a strong submission to a Scientific Journal when I can find the time to work more on this.

1 Introduction to Clique

In computer science, the clique problem is the computational problem of finding cliques (subsets of vertices, all adjacent to each other, also called complete subgraphs) in a graph. It has several different formulations depending on which cliques, and what information about the cliques, should be found. Common formulations of the clique problem include finding a maximum clique (a clique with the largest possible number of vertices), finding a maximum weight clique in a weighted graph, listing all maximal cliques (cliques that cannot be enlarged), and solving the decision problem of testing whether a graph contains a clique larger than a given size.

The clique problem arises in the following real-world setting. Consider a social network, where the graph's vertices represent people, and the graph's edges represent mutual acquaintance. Then a clique represents a subset of people who all know each other, and algorithms for finding cliques can be

used to discover these groups of mutual friends. Along with its applications in social networks, the clique problem also has many applications in bioinformatics and computational chemistry.

Most versions of the clique problem are hard. The clique decision problem is NP-complete (one of Karp's 21 NP-complete problems). The problem of finding the maximum clique is stated to be both fixed-parameter intractable and hard to approximate. Most experts in the field have concluded that listing all maximal cliques may require exponential time as there exist graphs with exponentially many maximal cliques.

Most, if not all experts in the field agree that to find a maximum clique, one can systematically inspect all subsets, but this sort of brute-force search is too time-consuming to be practical for networks comprising more than a few dozen vertices. Although (prior to the present paper), no polynomial time algorithm is known for this problem, more efficient algorithms than the brute-force search are known. For instance, the Bron–Kerbosch algorithm can be used to list all maximal cliques in worst-case optimal time, and it is also possible to list them in polynomial time per clique.

While a method for computing the solutions to NP-complete problems using a reasonable amount of time has remained undiscovered (until the publication of the present paper), computer scientists and programmers still frequently encounter NP-complete problems. NP-complete problems are often addressed by using heuristic methods and approximation algorithms. The present author feels that this is a poor methodology and strategy.

The ability to compute solutions for problems such as clique (and many others NP-Complete problems) has been deemed the holy grail of computational complexity theory and is one of the Millennium Prize Problems.

2 Description of the New Approach

The present paper provides a novel approach to solving the clique problem(s). The present methods will work for any clique problem, including those which are determined to be NP-Complete. Determining other cliques, such as cliques of a fixed size (k=3, k=4, etc.) is trivial by comparison but will also be described.

The author provides a means for greatly reducing the time that it will take a computer (or human) to solve for:

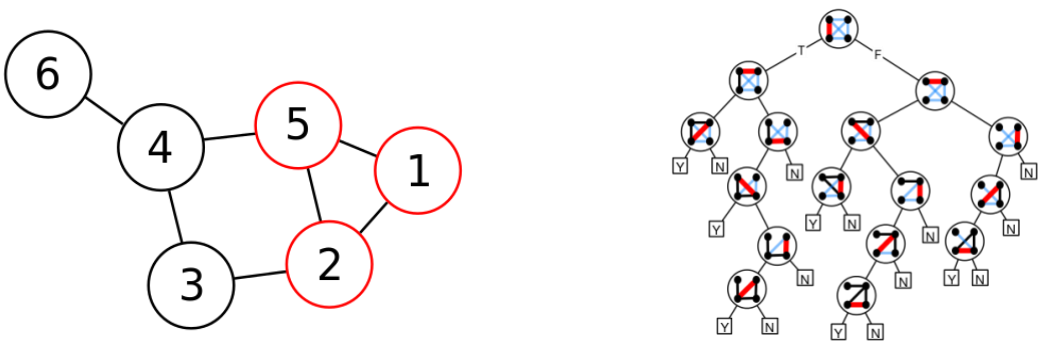
1. Maximum clique (a clique with the largest possible number of vertices),
2. Listing all maximal cliques (cliques that cannot be enlarged), and
3. Solving the decision problem of testing whether a graph contains a clique larger than a given size.

To solve the clique problem, the author feels that we must discard previous graphing methods and start from scratch with a new strategy.

Traditional methods of organizing data don't seem to work for sorting data into cliques or for finding a maximum clique in polynomial time. The present novel method requires that the data and each of its variables be converted into one of two binary systems so that each and every permutation can be accounted for and compared to each other in more logical way. In our new method each variable will take on a single binary place value such as 1, 2, 4, 8, 16, 32, 64 etc.

By organizing our data this way, we can account for every possible permutation that might occur, no matter what the size of the input, without requiring brute-force or exponential time strategies. The present innovation requires utilization of (potentially two different) Boolean Incidence Matrices.

Figure 1



Present methods of working with graphs as seen above (Figure 1), are one of the problems why finding items, such as a largest clique(s) is a real problem for computational devices. Attempting to solve a clique problem of any significant size by utilizing these methods is not feasible.

Mathematicians generally order elements of clique in an ordered progression 1, 2, 3, 4, 5, 6, etc., but herein lies the problem when trying to find specific size cliques or groups within the larger structure. The present invention matches a binary place valued number to the variables, which in this example are names prior to computation.

As an example;

John becomes 1

Sue becomes 2

Bob becomes 4

Jenn becomes 8

Colin becomes 16

Maggy becomes 32

Jim becomes 64

Kelly becomes 128

Figure 2

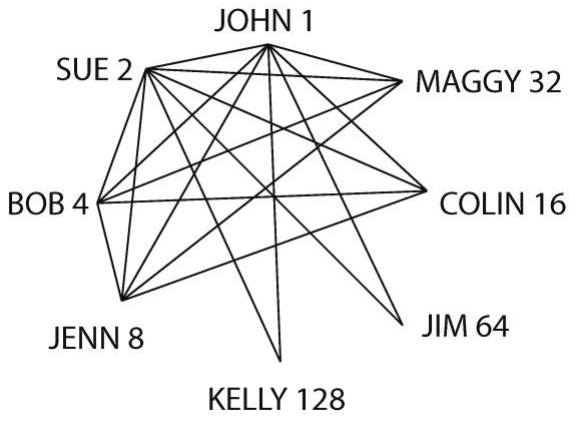


Figure 3

CLIQUE		Kelly	Jim	Maggy	Colin	Jenn	Bob	Sue	John	
		128	64	32	16	8	4	2	1	
1	John	1	1	1	1	1	1	1	1	**255**
2	Sue	1	1	1	1	1	1	1	1	**255**
4	Bob	0	0	1	1	1	1	1	1	**63**
8	Jenn	0	0	1	1	1	1	1	1	**63**
16	Colin	0	0	0	1	1	1	1	1	**31**
32	Maggy	0	0	1	0	1	1	1	1	**47**
64	Jim	0	1	0	0	0	0	1	1	**67**
128	Kelly	1	0	0	0	0	0	1	1	**131**
		131	**67**	**47**	**31**	**63**	**63**	**255**	**255**	

The above diagrams (Figures 2 & 3) provides an example for one instance of clique. It is a very simple version of clique with a very small input of only eight individuals. Most instances or examples of clique will have many more variables to be sorted, but for simplicity we will begin with this basic example.

Next we need to utilize what I call "Collaboritive Variables",.

Note: I initially outlined and described in-depth in a paper that I published on Research Gate on ___date____ for TSP and Sudoku. In hind sight I should have also utilized Collaborative Variables' in clique as well. Blunders happen more easily when nobody will proof read your papers I suppose.

In the provided example (Figure 3), John has been assigned the binary variable 1 and Sue 2. John and Sue are then further assigned the number 255. 255 shows their relationship to themselves, each other and everybody else in the data set.

By following this logic, we can also see that IF an individual is assigned a number 254, 253, 251, 247, 239, 223, 191 or 127, then they will be friends with six people, themselves and enemies with one. We can see this clearly in the binary representations. 11111110, 1111101, 11111011, 11110111, 11101111, 11011111, 10111111, 01111111. These binary numbers represent both themselves and all their friendships within the data set. A high or low number doesn't mean that you have more or less friends,

but it instead tells us exactly who your friends are. This will be useful when we begin sorting and filtering for requested outputs.

Figure 4 below provides a bell curve for the distribution of binary digits. Numbers with four binary digits out of eight turned ON are the most frequent). Numbers with ALL ON or ALL OFF are the least frequent (there is only one option for zero **ALL OFF** and only one option for 255 **ALL ON**). The importance of this will be explained. Cliques of different sizes will obviously have different distributions and sometimes different curves.

Figure 4

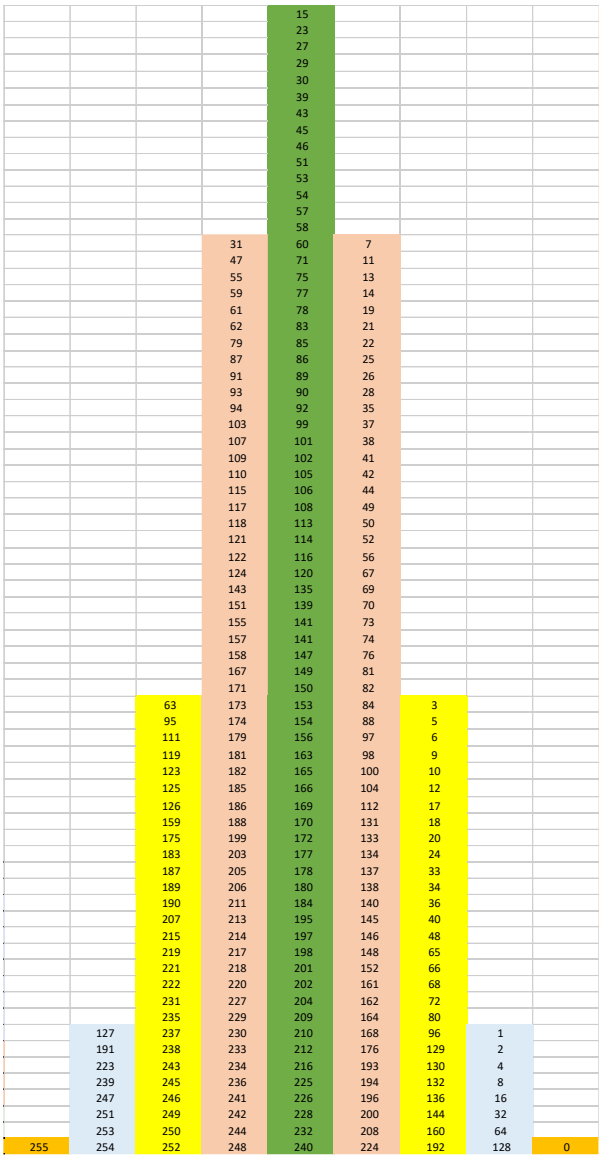
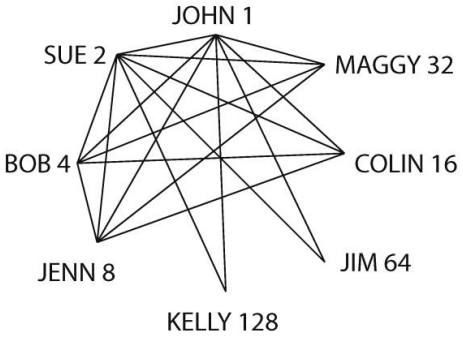


Figure 2 (Again)



Eureka!

My system requires the use of collaborative variables (as were initially outlined in my 1st P=NP paper entitled Methods for Organizing Data July 2018), but I forgot to put them in my later Clique papers. The TSP and Sudoku explanations are ok. Collaborative Variables may very well be the key to the P = NP Universe.

For an 8-Clique we know that there are 256 possible variations to search --- (in the worst-case scenario). Computers will presently search them all, which is trivial for an 8-CLIQUE, however it could well take millions of years for a 1000-CLIQUE problem.

HOW?

Our 1st step is to sort our data as a 1 by 1 matrix as seen again below.

CLIQUE		Kelly	Jim	Maggy	Colin	Jenn	Bob	Sue	John	
		128	64	32	16	8	4	2	1	
1	John	1	1	1	1	1	1	1	1	**255**
2	Sue	1	1	1	1	1	1	1	1	**255**
4	Bob	0	0	1	1	1	1	1	1	**63**
8	Jenn	0	0	1	1	1	1	1	1	**63**
16	Colin	0	0	0	1	1	1	1	1	**31**
32	Maggy	0	0	1	0	1	1	1	1	**47**
64	Jim	0	1	0	0	0	0	1	1	**67**
128	Kelly	1	0	0	0	0	0	1	1	**131**
		131	**67**	**47**	**31**	**63**	**63**	**255**	**255**	

Figure 4

	Kelly	Jim	Maggy	Colin	Jenn	Bob	Sue	John	
	128	64	32	16	8	4	2	1	
John & Kelly	128	0	0	0	0	0	2	1	**131**
John & Jim	0	64	0	0	0	0	2	1	**67**
John & Maggy	0	0	32	0	8	4	2	1	**47**
John & Colin	0	0	0	16	8	4	2	1	**31**
John & Jenn	0	0	32	16	8	4	2	1	**63**
John & Bob	0	0	32	16	8	4	2	1	**63**
John & Sue	128	64	32	16	8	4	2	1	**255**
Sue & Kelly	128	0	0	0	0	0	2	1	**131**
Sue & Jim	0	64	0	0	0	0	2	1	**67**
Sue & Maggy	0	0	32	0	8	4	2	1	**47**
Sue & Colin	0	0	0	16	8	4	2	1	**31**
Sue & Jenn	0	0	32	16	8	4	2	1	**63**
Sue & Bob	0	0	32	16	8	4	2	1	**63**
Bob & Kelly	0	0	0	0	0	0	2	1	**3**
Bob & Jim	0	0	0	0	0	0	2	1	**3**
Bob & Maggy	0	0	32	0	8	4	2	1	**47**
Bob & Colin	0	0	0	16	8	4	2	1	**31**
Bob & Jenn	0	0	32	16	8	4	2	1	**63**
Jenn & Kelly	0	0	0	0	0	0	2	1	**3**
Jenn & Jim	0	0	0	0	0	0	2	1	**3**
Jenn & Maggy	0	0	32	0	8	4	2	1	**47**
Jenn & Colin	0	0	0	16	8	4	2	1	**31**
Colin & Kelly	0	0	0	0	0	0	2	1	**3**
Colin & Jim	0	0	0	0	0	0	2	1	**3**
Colin & Maggy	0	0	0	0	8	4	2	1	**15**
Maggy & Kelly	0	0	0	0	0	0	2	1	**3**
Maggy & Jim	0	0	0	0	0	0	2	1	**3**
Jim & Kelly	0	0	0	0	0	0	2	1	**3**

We then use our collaborative Variables which could be likened to a 2 by 1 Matrix!

We actually only need to do collaborative variable scenarios when each pairing of two people are FRIENDS. In red we see the NOT FRIENDS, which we did not need to even look at! Below I have adjusted the table and deleted them.

Figure 5

	Kelly	Jim	Maggy	Colin	Jenn	Bob	Sue	John	
	128	64	32	16	8	4	2	1	
John & Kelly	128	0	0	0	0	0	2	1	**131**
John & Jim	0	64	0	0	0	0	2	1	**67**
John & Maggy	0	0	32	0	8	4	2	1	**47**
John & Colin	0	0	0	16	8	4	2	1	**31**
John & Jenn	0	0	32	16	8	4	2	1	**63**
John & Bob	0	0	32	16	8	4	2	1	**63**
John & Sue	128	64	32	16	8	4	2	1	**255**
Sue & Kelly	128	0	0	0	0	0	2	1	**131**
Sue & Jim	0	64	0	0	0	0	2	1	**67**
Sue & Maggy	0	0	32	0	8	4	2	1	**47**
Sue & Colin	0	0	0	16	8	4	2	1	**31**
Sue & Jenn	0	0	32	16	8	4	2	1	**63**
Sue & Bob	0	0	32	16	8	4	2	1	**63**
Bob & Maggy	0	0	32	0	8	4	2	1	**47**
Bob & Colin	0	0	0	16	8	4	2	1	**31**
Bob & Jenn	0	0	32	16	8	4	2	1	**63**
Jenn & Maggy	0	0	32	0	8	4	2	1	**47**
Jenn & Colin	0	0	0	16	8	4	2	1	**31**

I then sort the answers and we get Figure 6

Figure 6

	Kelly	**Jim**	**Maggy**	**Colin**	**Jenn**	**Bob**	**Sue**	**John**	
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	
John & Sue	128	64	32	16	8	4	2	1	**255**
John & Kelly	128	0	0	0	0	0	2	1	**131**
Sue & Kelly	128	0	0	0	0	0	2	1	**131**
John & Jim	0	64	0	0	0	0	2	1	**67**
Sue & Jim	0	64	0	0	0	0	2	1	**67**
John & Jenn	0	0	32	16	8	4	2	1	**63**
John & Bob	0	0	32	16	8	4	2	1	**63**
Sue & Jenn	0	0	32	16	8	4	2	1	**63**
Sue & Bob	0	0	32	16	8	4	2	1	**63**
Bob & Jenn	0	0	32	16	8	4	2	1	**63**
John & Maggy	0	0	32	0	8	4	2	1	**47**
Sue & Maggy	0	0	32	0	8	4	2	1	**47**
Bob & Maggy	0	0	32	0	8	4	2	1	**47**
Jenn & Maggy	0	0	32	0	8	4	2	1	**47**
John & Colin	0	0	0	16	8	4	2	1	**31**
Sue & Colin	0	0	0	16	8	4	2	1	**31**
Bob & Colin	0	0	0	16	8	4	2	1	**31**
Jenn & Colin	0	0	0	16	8	4	2	1	**31**

The left most Column provides us with all of our Maximal Cliques. Obviously smaller cliques can be made from these, but the heavy lifting has been done. This hard problem has become EASY.

The true beauty of my system is that it scales absolutely wonderfully!

Here is a quick 11 variable version!

Pretend that the letters are friends and/or enemies.

1 denotes a friend in this instance.

Figure 7

	K	J	I	H	G	F	E	D	C	B	A
A	1	0	1	0	1	0	1	0	1	1	1
B	0	0	0	0	0	0	1	0	0	1	1
C	1	0	1	0	1	0	1	1	1	0	1
D	0	0	0	0	0	1	1	1	1	0	0
E	1	0	1	1	1	1	1	0	1	1	1
F	0	1	0	0	0	1	1	1	1	0	0
G	1	1	1	0	1	0	1	0	1	0	1
H	0	1	1	1	0	0	1	0	0	0	0
I	1	0	1	1	1	0	1	0	1	0	1
J	1	1	0	1	1	1	0	0	0	0	0
K	1	1	1	0	1	0	1	0	1	0	1

We then Collaborate the Variables that are FRIENDS

Figure 8

	K	J	I	H	G	F	E	D	C	B	A	
	1024	512	256	128	64	32	16	8	4	2	1	
A&B	0	0	0	0	0	0	16	0	0	2	1	19
A&C	1024	0	256	0	64	0	16	0	4	0	1	1365
A&E	1024	0	256	0	64	0	16	0	4	2	1	1367
A&G	1024	0	256	0	64	0	16	0	4	0	1	1365
A&I	1024	0	256	0	64	0	16	0	4	0	1	1365
A&K	1024	0	256	0	64	0	16	0	4	0	1	1365
B&E	0	0	0	0	0	0	16	0	4	2	0	22
B&E	0	0	0	0	0	0	16	0	0	2	1	19
C&D	0	0	0	0	0	0	16	8	4	0	0	28
C&E	1024	0	256	0	64	0	16	0	4	0	1	1365
C&G	1024	0	256	0	64	0	16	0	4	0	1	1365
C&I	1024	0	256	0	64	0	16	0	4	0	1	1365
C&K	1024	0	256	0	64	0	16	0	4	0	1	1365
D&F	0	0	0	0	0	32	16	8	4	0	0	60
E&F	0	0	0	0	0	32	16	0	4	0	0	52
E&G	1024	0	256	0	64	0	16	0	4	0	1	1365
E&H	0	0	256	128	0	0	16	0	0	0	0	400
E&I	1024	0	256	128	0	0	16	0	4	0	1	1429
E&K	1024	0	256	0	64	0	16	0	4	0	1	1365
F&J	0	512	0	0	0	32	0	0	0	0	0	544
G&I	1024	0	256	0	64	0	16	0	4	0	1	1365
G&J	1024	512	0	0	64	0	0	0	0	0	0	1600
G&K	1024	512	256	0	64	0	16	0	4	0	1	1877
H&I	0	0	256	0	0	0	16	0	0	0	0	272
H&J	0	512	0	128	0	0	0	0	0	0	0	640
J&K	1024	512	0	0	64	0	0	0	0	0	0	1600

Then we sort in Figure 9

Figure 9

	K	J	I	H	G	F	E	D	C	B	A	
	1024	512	256	128	64	32	16	8	4	2	1	
G&K	1024	512	256	0	64	0	16	0	4	0	1	1877
G&J	1024	512	0	0	64	0	0	0	0	0	0	1600
J&K	1024	512	0	0	64	0	0	0	0	0	0	1600
E&I	1024	0	256	128	0	0	16	0	4	0	1	1429
A&E	1024	0	256	0	64	0	16	0	4	2	1	1367
A&C	1024	0	256	0	64	0	16	0	4	0	1	1365
A&G	1024	0	256	0	64	0	16	0	4	0	1	1365
A&I	1024	0	256	0	64	0	16	0	4	0	1	1365
A&K	1024	0	256	0	64	0	16	0	4	0	1	1365
C&E	1024	0	256	0	64	0	16	0	4	0	1	1365
C&G	1024	0	256	0	64	0	16	0	4	0	1	1365
C&I	1024	0	256	0	64	0	16	0	4	0	1	1365
C&K	1024	0	256	0	64	0	16	0	4	0	1	1365
E&G	1024	0	256	0	64	0	16	0	4	0	1	1365
E&K	1024	0	256	0	64	0	16	0	4	0	1	1365
G&I	1024	0	256	0	64	0	16	0	4	0	1	1365
H&J	0	512	0	128	0	0	0	0	0	0	0	640
F&J	0	512	0	0	0	32	0	0	0	0	0	544
E&H	0	0	256	128	0	0	16	0	0	0	0	400
H&I	0	0	256	0	0	0	16	0	0	0	0	272
D&F	0	0	0	0	0	32	16	8	4	0	0	60
E&F	0	0	0	0	0	32	16	0	4	0	0	52
C&D	0	0	0	0	0	0	16	8	4	0	0	28
B&E	0	0	0	0	0	0	16	0	4	2	0	22
A&B	0	0	0	0	0	0	16	0	0	2	1	19
B&E	0	0	0	0	0	0	16	0	0	2	1	19

Figure 9 gives us all the Maximal Cliques larger than two. We can see that 1600 or GJ&K are friends, 19 or AB&E are friends, but that our Maximum Clique is ACGIK&E are all friends or a 6 clique! Yay!

In some instances, it may be beneficial to flip the bits and search for NOT FRIENDS rather than friends as it may require less data entry and search efforts. People may still be friends with themselves though and so we hit about a 50-50% mark….meaning if it seems that most people are friends we may be better served by searching the NOT FRIEND Cliques to save time and extrapolate FRIEND groups from the opposite data.

In some instances we may also want to utilize larger collaborative variables of 3, 4, 5 etc… as a group for various reasons.

Scaling and Worst Case Scenario

For an 8 clique we require 7+6+5+4+3+2+1 * 8 = 224 data entry points (for 256 possible permutations) as a worst case scenario if everyone are friends. (Obviously this is an error as we know we have an 8 by 8 clique as everyone would be friends with everyone). We also know that if very few folks are friends, we may convert to searching NOT FRIEND Cliques to save space, but let's simply continue for a fun example of worst case scenario!

For a 9 clique we would require 8+7+6+5+4+3+2+1 * 9 = 324 data entry points (for 512 possible permutations). Figure 10 below continues this train of thought. We can see that as the problem scales that even though it still grows, that it grows at a smaller and smaller rate! I also discussed this type of growth rate in my TSP paper.

Figure 10

8+7+6+5+4+3+2+1	Equals	36	*	9	Equals	324	data entry elements for	512	Permutations
9+8+7+6+5+4+3+2+1	Equals	46	*	10	Equals	460	data entry elements for	1,024	Permutations
10+9+8+7+6+5+4+3+2+1	Equals	57	*	11	Equals	627	data entry elements for	2,048	Permutations
etc….	Equals	69	*	12	Equals	828	data entry elements for	4,096	Permutations
etc….	Equals	82	*	13	Equals	1,066	data entry elements for	8,192	Permutations
etc….	Equals	95	*	14	Equals	1,330	data entry elements for	16,384	Permutations
etc….	Equals	108	*	15	Equals	1,620	data entry elements for	32,768	Permutations
etc….	Equals	121	*	16	Equals	1,936	data entry elements for	65,536	Permutations
etc….	Equals	134	*	17	Equals	2,278	data entry elements for	131,072	Permutations
etc….	Equals	147	*	18	Equals	2,646	data entry elements for	262,144	Permutations
etc….	Equals	160	*	19	Equals	3,040	data entry elements for	524,288	Permutations
etc….	Equals	173	*	20	Equals	3,460	data entry elements for	1,048,576	Permutations
etc….	Equals	186	*	21	Equals	3,906	data entry elements for	2,097,152	Permutations
etc….	Equals	199	*	22	Equals	4,378	data entry elements for	4,194,304	Permutations
etc….	Equals	212	*	23	Equals	4,876	data entry elements for	8,388,608	Permutations
etc….	Equals	225	*	24	Equals	5,400	data entry elements for	16,777,216	Permutations
etc….	Equals	238	*	25	Equals	5,950	data entry elements for	33,554,432	Permutations
etc….	Equals	251	*	26	Equals	6,526	data entry elements for	67,108,864	Permutations
etc….	Equals	264	*	27	Equals	7,128	data entry elements for	134,217,728	Permutations
etc….	Equals	277	*	28	Equals	7,756	data entry elements for	268,435,456	Permutations
etc….	Equals	290	*	29	Equals	8,410	data entry elements for	536,870,912	Permutations

https://jamesmccaffrey.wordpress.com/2011/06/24/the-maximum-clique-problem/

"It turns out that finding the maximum clique for graphs of even moderate size is one of the most challenging problems in computer science. The problem is NP-complete which means, roughly, that every possible answer must be examined. Suppose we have a graph with six nodes. First we'd try to see if all six nodes form a clique. There is Choose (6,6) = 1 way to do this. Next we'd examine all groups of five nodes at a time; Choose (6,5) = 6 ways. And so on, checking Choose (6,4) = 15, Choose (6,3) = 20, Choose (6,2) = 15, and Choose(6,1) = 6 possible solutions for a total of 63 checks. (For the maximum clique problem we can stop when we find the largest clique so let's assume that on average we'd have to go through about one-half of the checks).

The total number of checks increases very quickly as the size of the graph, n, increases. For n = 10 there are 1,023 total combinations. For n = 20 there are 1,048,575 combinations. But for n = 1,000 there are 10,715,086,071,862,673,209,484,250,490,600,018,105,614,048,117,055,336,074,437,503,883,703,510,511,249,361,224,931,983,788,156,958,581,275,946,729,175,531,468,251,871,452,856,923,140,435,984,577,574,698,574,803,934,567,774,824,230,985,421,074,605,062,371,141,877,954,182,153,046,474,983,581,941,267,398,767,559,165,543,946,077,062,914,571,196,477,686,542,167,660,429,831,652,624,386,837,205,668,069,375 combinations. Even if you could perform one trillion checks per second it would take you 3.4 x 10^281 years which is insanely longer than the estimated age of the universe (about 1.0 x 10^10 = 14 billion years)."

- James McCaffrey

The above is no longer true…

F - References

1. Cook, S.A. (1971). "The complexity of theorem proving procedures". Proceedings, Third Annual ACM Symposium on the Theory of Computing, ACM, New York. pp. 151–158. doi:10.1145/800157.805047.
2. Wikipedia contributors. (2018, June 19). Clique problem. In *Wikipedia, The Free Encyclopedia*. Retrieved 13:57, June 28, 2018, from https://en.wikipedia.org/w/index.php?title=Clique_problem&oldid=846513850

3. Wikipedia contributors. (2018, May 14). NP-completeness. In *Wikipedia, The Free Encyclopedia*. Retrieved 14:00, June 28, 2018, from https://en.wikipedia.org/w/index.php?title=NP-completeness&oldid=841292328
4. Gödel's Lost Letter and P=NP https://rjlipton.wordpress.com/the-gdel-letter/
5. Garey, Michael R.; David S. Johnson (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman. *ISBN 0-7167-1045-5*