

# Solving Maximal Clique Problem through Genetic Algorithm

Cite as: AIP Conference Proceedings **1324**, 235 (2010); <https://doi.org/10.1063/1.3526202>  
Published Online: 03 December 2010

Shalini Rajawat, Naveen Hemrajani and Ekta Menghani



View Online



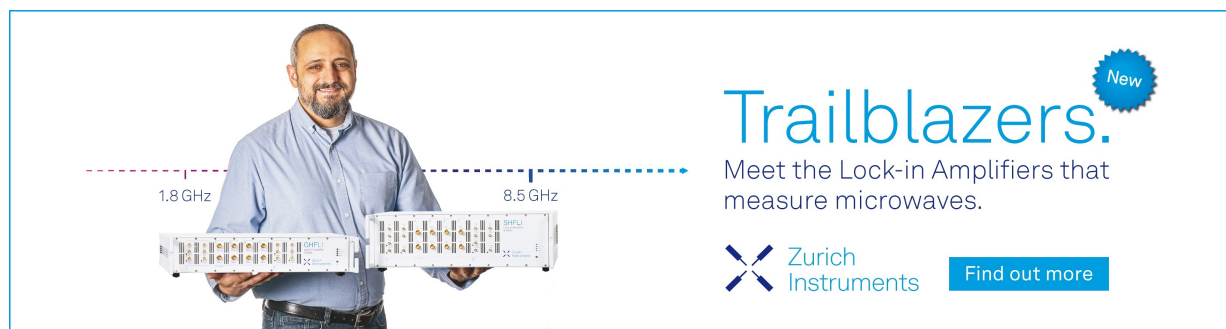
Export Citation

## ARTICLES YOU MAY BE INTERESTED IN

[Application of Bron-Kerbosch algorithm in graph clustering using complement matrix](#)  
AIP Conference Proceedings **1862**, 030141 (2017); <https://doi.org/10.1063/1.4991245>

[Solving TSP problem with improved genetic algorithm](#)  
AIP Conference Proceedings **1967**, 040057 (2018); <https://doi.org/10.1063/1.5039131>

[A survey of SAT solver](#)  
AIP Conference Proceedings **1836**, 020059 (2017); <https://doi.org/10.1063/1.4981999>



**Trailblazers.** New

Meet the Lock-in Amplifiers that measure microwaves.

Zurich Instruments [Find out more](#)

# Solving Maximal Clique Problem through Genetic Algorithm

\* Shalini Rajawat, \* Naveen Hemrajani and \*\* Ekta Menghani

\*Department of Computer Science and Engineering, Suresh Gyan Vihar University, Jaipur India

\*\*Department of Biotechnology, Mahatma Gandhi Institute of Applied Sciences, Jaipur, India

**Abstract** Genetic algorithm is one of the most interesting heuristic search techniques. It depends basically on three operations; selection, crossover and mutation. The outcome of the three operations is a new population for the next generation. Repeating these operations until the termination condition is reached. All the operations in the algorithm are accessible with today's molecular biotechnology. The simulations show that with this new computing algorithm, it is possible to get a solution from a very small initial data pool, avoiding enumerating all candidate solutions. For randomly generated problems, genetic algorithm can give correct solution within a few cycles at high probability.

**Keywords:** Random algorithm, genetic algorithm, NP-complete problem, DNA computing.

## I INTRODUCTION

Genetic algorithms and other types of "evolutionary computation" come in many, different varieties, but most are variations and/or elaborations on the following very loose outline. Genetic algorithms typically cycle electronic computers through the following steps, maintaining a population of bit strings representing candidate solutions of a problem. Genetic Algorithm: Begin with a diverse initial population, perhaps chosen randomly.

1. Evaluate fitness of candidates.
2. Select more fit candidates to breed and lesser candidates to be replaced.
3. Induce variation by breeding.

## II METHODS

In Figure 1a, a problem of five vertices and five edges is defined. Obviously, the vertices (2,3,4) make the largest clique, so the size of the largest clique in this network is three. The maximal clique problem has been proven an NP-complete problem. Besides the conventional algorithm on electronic computer and DNA computer, some interesting attempts have been made<sup>[15]</sup>.

The data structure of the computation is the same of the previous work<sup>[3]</sup>. For a graph of N vertices, we use an N-digit binary number to represent each possible clique. In the N-digit binary number, a bit set to 1 represents the corresponding vertex in the clique, a bit set to 0 represents the corresponding vertex out of the clique. For example, 5-digit binary number (01110)

represents the largest clique (2,3,4) in Figure 1a. To solve the maximal clique problem is to find an N- digit

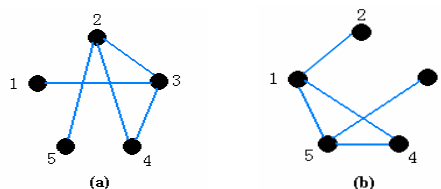


Figure 1(a): The original graph describing a maximal clique problem of five vertices and five edges; (b): complementary graph

binary number that contains most 1s (called condition A). For each graph of N vertices, we can build a complementary graph, which contains all the missing edges in the original graph, as shown in Figure 1b.

It should be noticed that if the complementary graph can be divided into several connected graphs, the problem can be divided into several sub-problems, find each sub-problem's maximal clique, and then the union of these cliques makes the whole problem's maximal clique. From this point of view, one need to solve problems whose complementary graph is a connected graph. Obviously, any two vertices in a clique in the original graph are *not* connected in the complementary graph. All the calculative processes can be readily mapped into corresponding biological operations using current biotechnologies.

Biotechnologies that need to carry out the algorithm. Encode the binary numbers into DNA strands, one strand representing one binary number. We noticed that the parallel overlap assembly technique of building the

data pool to be very helpful to our algorithm. To encode an N-digit binary number, we need N+1 DNA sequences representing the positions and N sequences representing the values of each digit. The sequences representing the positions are all of the same length, but the sequences representing the different values are of different length, for example, 1s are shorter than 0s. It should be noticed that the sequences must be of some different characters

The operation of elimination can be carried out with at least two techniques first, we make the length of the value sequences of 0s be zero, then wherever in a binary number 0 occurs, the two position sequences next to it will join. In order to eliminate the unqualified DNA strands, we only have to pick up the qualified ones by using polyacrylamide gel containing bound probes to capture the strands with 0 at any of the two positions (Figure 2b) [4].

For another, If we make the length of the value sequences of 1s be zero, then wherever in a binary number 1 occurs, the two position sequences next to that digit will be directly linked. If we design the position sequences in such a way that a restriction enzyme can break the linking point, we will eliminate value 1 in that specific position. If we want to break all the strands with 1 at position a and position b, we only need to divide the data pool into two parts, pool A and pool B, breaking the 1 at position a in pool A and 1 at position b in pool B. The union of pool A and pool B will not contain DNA strands with 1 at both positions but still can contain strands with 1 at only one of the two positions (Figure 2a) [3].

To start with we take a small data pool containing many N-digit binary numbers. In the initial data pool, each digit of each binary number is set to 0, meaning that no vertex is in the clique. Then let the data pool evolve. In each cycle of evolution, first randomly replace some digits in the data pool by 1s with certain probability, no matter whether the original digit is 0 or 1. This operation is called *mutation*. Then *eliminate* from the data pool all numbers that do not satisfy condition A. For example, for the problem shown in Figure 1a and Figure 1b, such numbers as (11xxx), (1xx1x), (1x1xx), (xx1x1) and (xxx11) will be removed from the data pool, while x representing either 0 or 1. If the size of the data pool is smaller after the elimination, *enlarge* it by making copies of the binary numbers until the size of the pool reaches that of the initial pool. Through these operations of mutation and elimination, the cliques in the data pool will become bigger as the increase of the number of evolution cycle.

In order to find out if there is some progress in each cycle, after the operation of elimination, we will find the maximum number of 1s in a string. This number represents the

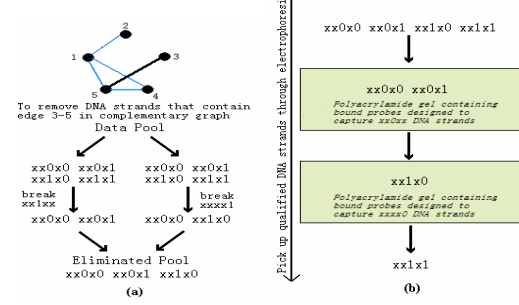


Figure 2(a) Eliminating the data pool with restricted enzymes;(b) eliminating the data pool by picking up qualified strands with probes in electrophoresis

current biggest clique's size. If this size tends to increase with the increase of number of cycles, it is known that the maximal clique is not found, so the computation should go on. If it stops growing for many cycles, we regard it as a sign that the evolution has reached its end, so the computation should stop and the size of the clique we get at last is considered as the size of the maximal clique.

### III RESULTS

The genetic algorithm can solve the problem correctly; the cycle needed to get the correct answers is almost a linear function of the number of nodes (See Figure 3(a). If the connecting rate of problems is fixed, then the number of edges in the complementary graph is order of  $N^2$ . In each cycle, each edge in the complementary

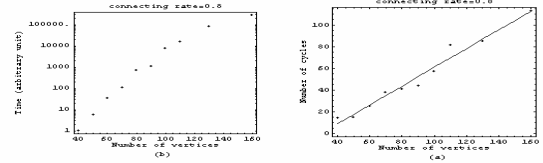


Figure 3(a): Comparison of related time requirement between GA and RA in solving MC problems with different sizes. The time requirement of the GA increases linearly;(b) The time requirement of RA increases exponentially

graph brings in one operation of elimination, if the number of cycles required increases linearly to N, then the total time requirement of our genetic algorithm is order of  $N^3$ . On the other hand, the total time requirement of the recursive algorithm is exponential to N, as shown in Figure 3(b).

In order to evaluate the algorithm, we randomly generated some problems of different sizes, to study the time requirements of our algorithm for problems of

different difficulty levels. The problems are generated with a connecting rate of approximate 80%, that is, when generating the problem, each pair of vertices has a probability of 80% to be connected by an edge. In our simulation, the size of data pool is  $10^6$ . The probability at which we put 1s into digits (the mutation rate) is  $2/N$  in the sense that the cliques will grow averagely at the rate of two vertices per cycle. The computation will stop if the size of the biggest clique remains unchanged for 30 cycles. We also use a conventional recursive algorithm (RA) to solve the problems in an exhaustive way in order to check the result of the genetic algorithm and to compare the time requirement of the two algorithms. Some results of our computation are shown in Table 1.

Vertices (N)	Edges	Size of MC	Time (Arbitrary unit, by RA)	Prob. to get the correct answer (by GA)	Average number of cycles used before reaching an answer
40	624	14	1	100% (6 of 6)	14
60	1420	16	34.2	100% (6 of 6)	25
80	2530	19	703	100% (6 of 6)	41
100	4000	20	7440	100% (6 of 6)	57
130	6700	21	80800	100% (6 of 6)	85
160	10000	22	281000	83% (5 of 6)	113

Table 1 conventional recursive algorithm (RA) to solve the problems in an exhaustive and check the result of the genetic algorithm with comparison of time requirement of the two algorithms.

The difficulty level of a problem is related not only to the number of vertices in the graph, but also to the connecting rate. From Table 1 we can see that at the connecting rate of 80%, the size of maximal clique increases very slowly with the increase of the number of vertices. This implies that for randomly generated problems, a connecting rate of 80% is still not high enough to give large cliques. Therefore, we also studied MC problems with fixed number of vertices (40 and 70) but with increasing number of edges. The main results of the computation are in Table 2. We observe that the size of maximal clique increases rapidly as the connecting rate approaches 100%. As the number of edges increases, the recursive algorithm

requires exponentially increasing time, as shown in Figure 4(c), 4(d). The number of edges in complementary graph decreases as the increase of number of edges in the original graph, which reduces the number of operations of elimination in each cycle, thus the increase of the total number of operations would be

Vertices (N)	Edges	Size of MC	Time (Arbitrary unit, by RA)	Cycles needed (by GA)
40 (Full Edges: 780)	700	19	38	15
	720	21	173	16
	760	27	15700	17
70 (Full Edges: 2415)	1950	18	268	38
	2050	20	1430	40
	2150	24	26800	44

Table 2

even smaller. To our surprise, the number of cycles needed for the genetic algorithm to solve the problem does not increase much, as shown in Fig 4(a), 4(b).

#### IV CONCLUSIONS

Because the maximal cliques have most vertices, they are easier to be finally arrived at than any other smaller cliques are. Although there are much more smaller cliques than the maximal cliques, most of them are on the right track to the maximal cliques. To make this point more clear,

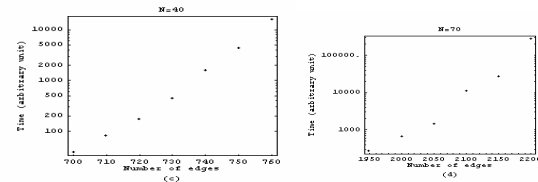


Figure 4(c) (d): Comparison of related time requirement between genetic algorithm and recursive algorithm in solving MC problems with different number of edges. The time requirement of recursive algorithm increases exponentially

see Figure 5, which shows the process of solving an 8-vertex problem. Figure (a) describes the problem, whose linking rate is 0.8. Figure (b) shows all the cliques, each as a point, divided into five levels, and all

the mutations that give larger cliques, each as a line. (For the reason of space, the exact clique that each point represents is not shown.) From bottom to top, the size of the cliques in each level is from 0 to 4. They grow bigger along the lines. The mutations that fail to give a clique are eliminated by selection and so are not contained in the graph. Notice that in level K, each point has K lines connecting to the points in level K-1, which means there are K ways to get a clique of size K represent those mutations and cliques that surely from

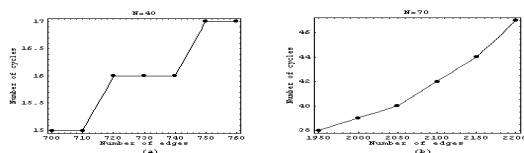


Figure 4 (a) (b): Comparison of related time requirement between GA and RA in solving MC problems with different number of edges. The time requirement of the genetic algorithm is almost unchanged

cliques of size K-1. The faded lines and points cannot lead to or end up as a maximal clique. As mentioned above, in randomly generated problems, they are only a small part of all mutations and cliques, the rest are on the right track. As long as percentage of cliques on the right track in each level is much bigger than  $1/D$ , where D is the size of the data pool, the genetic algorithm can almost surely give the correct solution after some cycles. Since the data pool of DNA molecules is very large, the capacity of the algorithm can be satisfying.

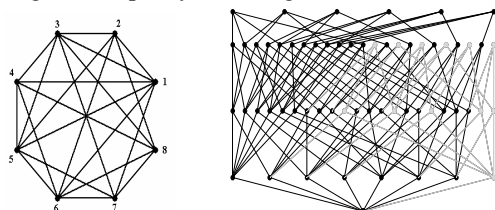


Figure 5 (a)

Figure 5(b)

Figure 5: (a), a problem of eight vertices, (b), A network describing the relation between all the cliques. Each vertex represents a different clique in (a). They are arranged in such a way that the sizes of the cliques are from 0 to 4 from bottom to top. Mutation can make cliques grow larger along the lines. The faded vertices and lines mean that the corresponding cliques and mutations cannot lead to the maximal cliques, which are at the top level.

## REFERENCES

- [1] Ruben, A. J. and Landweber, L. F., The past, present and future of molecular computing. *Nature Reviews Molecular Cell Biology*, 2000, 1: 69-72.
- [2] Adleman, L., Molecular computation of solutions to

- combinatorial problems. *Science*, 1994, 266: 1021-1024.
- [3] Ouyang, Q., Liu S. and Libchaber, A., DNA solution of the maximal clique problem. *Science*, 1997, 278: 446-449.
- [4] Braich, R. S., Chelyapov, N., Johnson, C., Rothermund, P. W. K. and Adleman, L., Solution of a 20-variable 3-SAT problem on a DNA computer. *Science*, 2002, 296: 499-502.
- [5] Faulhammer, D., Cukras, R. J. and Landweber, L. F., Molecular computation: RNA solutions to chess problems. *Proc. Natl. Acad. Sci.*, 2000, U.S.A. 97: 1385-1389.
- [6] Benenson, Y., Paz-Elizur, T., Adar, R., and Shapiro, E., Programmable and autonomous computing machine made of biomolecules. *Nature*, 2001, 414: 430-434.
- [7] Liu, Q., Wang, L., A. E., Corn, R. M. and Smith, L. M., DNA computing on surfaces. *Nature*, 2000, 403: 175-179.
- [8] Ogihara, M. and Ray, A., DNA computing on a chip. *Nature*, 2000, 403: 143-144.
- [9] Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T. and Hagiya, M., Molecular computation by DNA hairpin formation. *Science*, 2000, 288: 1223-1226.
- [10] Wang, L., Hall, J. G., Lu, M., Liu, Q. and Smith, L. M., A DNA computing readout operation based on structure-specific cleavage. *Nat. Biotechnol.*, 2001, 19: 1053-1059.
- [11] Zimmermann, Karl-Heinz, On applying molecular computation to binary linear codes. *IEEE Trans. Inform. Theory*, 2002, 48: 505-510.
- [12] Impagliazzo, R., Paturi, R. and Zane, F., Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 2001, 23: 512-530, doi: 10.1006/jcss.2001.1774.
- [13] Holland, J. H., Generic algorithm, *Scientific American*, 1992, July, 66-72.
- [14] Foster, J. A., Evolutionary computation. *Nat. Rev. Genet.*, 2001, 2: 428-436.
- [15] Chiu, D. T., Pezzoli, E., Wu, H., Stroock, A. D. and Whitesides, G. M., Using three-dimensional microfluidic networks for solving computationally hard problems. *Proc. Natl. Acad. Sci. U.S.A.*, 2001, 98: 2961-2966.
- [16] Kaplan, P. D., Ouyang, Q., Thaler, D. S. and Libchaber, A., Parallel overlap assembly for the construction of computational DNA libraries. *J. Theor. Biol.*, 1997, 188: 333-341, doi: 10.1006/jtbi.1997.0475.
- [17] Arita, M. and Kobayashi, S., The power of sequence design in DNA computing. *ICCIMA 2001: Fourth International Conference on Computational Intelligence and Multimedia Applications*, Proceedings, 163-167.