

Ramsey Numbers: Improving the Bounds of $R(5,5)$

Date: April 25, 2003

Curtis J. Kunkel
Department of Mathematics
University of Minnesota, Morris
Morris, MN 56267, kunk0003@mrs.umn.edu

Faculty Advisor: Prof. Peh Ng
Department of Mathematics
University of Minnesota, Morris
Morris, MN 56267, pehng@mrs.umn.edu

Abstract:

A Ramsey Number $r = R(m,n)$ is the smallest integer r such that a graph of r vertices has either a complete subgraph (clique) of size m or its complement has a complete subgraph of size n (independent set). Currently, the exact value of $R(5,5)$ is unknown, however it is known to be bounded by $43 \leq R(5,5) \leq 49$. In this paper we will discuss an attempt to discover a better lower bound by using Genetic Algorithms (GA). Our method involved standard genetic algorithm mutation and one-point crossover as recombination techniques. Results include graphs that were closer to being a counterexample than a random search technique achieved. A counterexample is a graph of size 43 that would prove 43 is too small and thus increase the lower bound to 44. Such a graph was not found.

1. Introduction to Problem

A graph, denoted $G = (V, E)$, consists of a set of vertices, V , and a set of edges, E , where the edges are lines that connect pairs of vertices. For example, Figure 1 (a), (b), (c), and (d) are examples of graphs. A graph is complete if every vertex is connected to all other vertices by an edge. Figure 1 (a) is a complete graph with 5 vertices. For any graph, G , its complement, \bar{G} , is another graph with the same set of vertices, where edges exist between two vertices in \bar{G} if and only if the edge does not exist between the same two vertices in G . Figure 1 (b) and (d) illustrate the differences between a graph and its complement. Let m, n be integers, the Ramsey Number, $v = R(m, n)$, is the minimum number v , such that all graphs of size v contain a complete subgraph of size m in G or a complete subgraph of size n in \bar{G} .

Example 1: A Ramsey Number, $R(m, n) = r$, is the smallest number r such that every group of r people at a party has m people who know each other or n people who do not know anyone else in their group of n people.

This gives us a good base to start from, but when we think of this in a little more mathematical way, it can be defined the following:

Definition 1: A Ramsey Number, $r = R(m, n)$, is the smallest number r , such that every graph of size r has either a clique of size m or an independent set of size n .

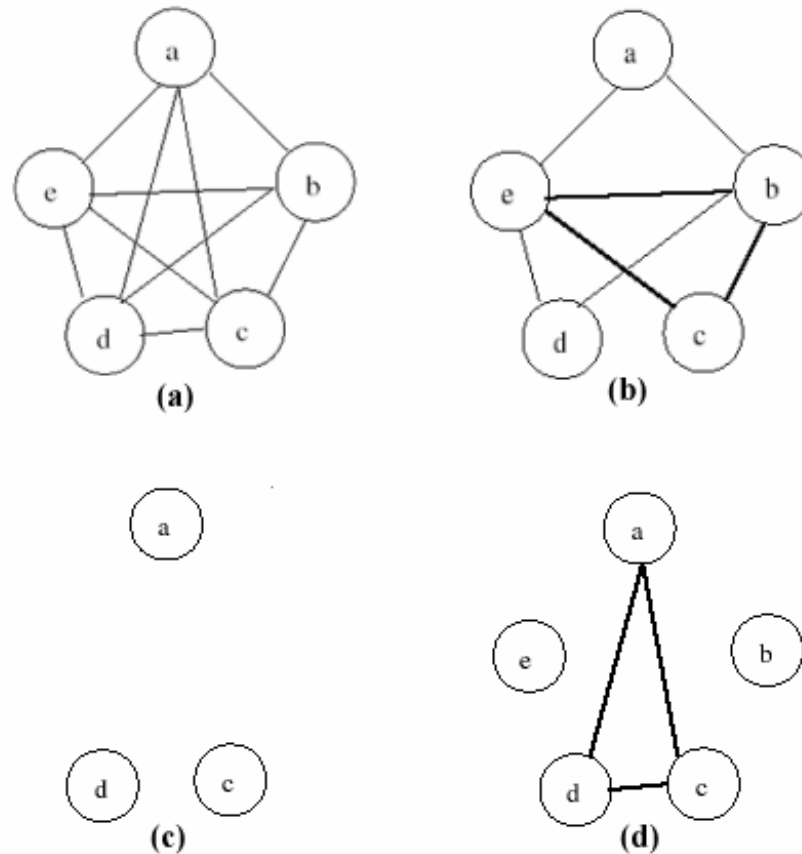
This also has an equivalent definition, given by the following:

Definition 2: A Ramsey Number, $r = R(m, n)$, is the smallest number r , such that every graph of size r has either a clique of size m in the graph or a clique of size n in its complement.

The main goal of this project is the optimization of $r = R(5, 5)$, i.e. finding the smallest integer r such that all graphs, G , of size r include a clique of size 5 in either G or the complement of G . Currently, $R(5, 5)$ is bounded by $43 \leq R(5, 5) \leq 49$. We are going to focus on the current lower bound, 43.

In the following sections of this paper, graph theory concepts will be defined so as to better understand the aforementioned definitions of Ramsey Numbers. A general overview of Ramsey Theory will also be explored, along with some examples of similar Ramsey Numbers and their values/bounds. The implementation of our approach will be described next. The final sections of the paper contain results and their analysis, along with some conclusions that can be drawn from those results.

Figure 1: Graphs



2. Mathematical Foundation

This section will give a brief description of the main graph theory concepts used in Ramsey Numbers. We will discuss a few theories related to Ramsey Numbers and in particular, $R(5,5)$. For more information on graph theory, see [3], [8], and [9].

2.1. Graph Theory Foundation

In order to better understand Ramsey Numbers and how they relate to our project, we need to look at some basic concepts of graph theory. It is clear from Definition 1 that we need to define a graph, a clique, and an independent set before we can fully understand the definition of a Ramsey Number.

Definition 3: A simple graph, denoted $G = (V, E)$, consists of a set of vertices, V , and a set of edges, E , where the edges are lines that connect pairs of vertices. Figure 1 (a), (b), (c), and (d) show examples of graphs.

Definition 4: Given a simple graph $G = (V, E)$, a *clique* of G is a subset S of V such that the subgraph induced by S has the property that edges exist between all possible pairs of

vertices in S . Figure 1 (b) is an example of a clique of size 3 in a graph (the clique is in bold). $S = \{b,c,e\}$.

Definition 5: Given a simple graph $G = (V,E)$, an *independent set* of G is a subset S of V such that the subgraph of G induced by S has the property that no edges exist between any pair of vertices in S . Figure 1 (c) is an example of an independent set of size 3 of the graph in Figure 1 (b). $S = \{a,c,d\}$.

Definition 6: A *complete graph* is a graph such that all pairs of vertices have an edge that connects both vertices. The notation for complete graphs is K_i , where i is the number of vertices. Figure 1 (a) is an example of the complete graph K_5 .

Definition 7: The *complement* of graph $G = (V, E)$, denoted $\bar{G} = (V, \bar{E})$, is a graph with the same set of vertices V such that $\bar{E} = \{(u,v) : u,v \in V \text{ and } (u,v) \notin E\}$. For instance, Figure 1 (b) and Figure 1 (d) is an example of a graph and its complement.

The following result follows directly from Definition 7.

Result: Given G and its complement, \bar{G} , then $G \approx \bar{\bar{G}}$.

Now that we have some basic concepts of graph theory defined, we can proceed to show the equivalence of Definition 1 and Definition 2.

The relationship between a graph G and its complement \bar{G} as it relates to cliques and independent sets is as follows: if a subgraph forms a clique in G , then the vertices in this subgraph induces an independent set in the complement of G , \bar{G} . This makes our case, $R(5,5)$, a bit easier to look at. $R(5,5)$ is therefore defined as the smallest size of a graph G such that there exists a clique of size 5 in its graph or its complement.

2.2. Ramsey Theory Foundation

The old joke that is often associated with Ramsey Numbers is that if an alien spaceship were to come to Earth and demand that we tell them the answer to $R(5,5)$ or they will kill us all, it would take all the computing power in the world to find the answer for the aliens. If he asks for $R(6,6)$, we should try to kill them. This shows how difficult this problem is. Many people have been working on $R(5,5)$ for decades and have made some remarkable headway. This section is broken up into two different parts: ‘Upper Bound Theory’ and ‘Examples of Similar Ramsey Numbers’. ‘Upper Bound Theory’ will introduce some theory used to determine upper bounds of Ramsey Numbers. ‘Examples of Similar Ramsey Numbers’ will look at $R(3,3)$ and $R(7,7)$ as they relate to $R(5,5)$.

2.2.1. Upper Bound Theory

The following theorems yield different ways of determining upper bounds of Ramsey Numbers. Erdős and Szekeres are given credit for proving Theorem 1. [5]

$$\textit{Theorem 1: } R(m,n) \leq \binom{m+n-2}{m-1}$$

Proof: We proceed by induction on k , where $k = m + n$. There is equality when $m = 1$ and $m = 2$, and for every value of n . It is also true for $n = 1$ and $n = 2$, independent of the value of m . Therefore, the result is true for all values of k with $2 \leq k \leq 5$ and we proceed through the remainder of the proof assuming that $m \geq 3$ and $n \geq 3$.

Assume that $R(s,t)$ exists for all positive integers s and t with $s + t < k$ where $k \geq 6$, and that

$$R(s,t) \leq \binom{s+t-2}{s-1}$$

Let m and n be integers such that $m + n = k$, where $m \geq 3$ and $n \geq 3$. By the inductive hypothesis, the Ramsey numbers $R(m-1,n)$ and $R(m,n-1)$ exist and further,

$$R(m-1,n) \leq \binom{m+n-3}{m-2} \quad \text{and} \quad R(m,n-1) \leq \binom{m+n-3}{m-1}$$

Since

$$\binom{m+n-3}{m-2} + \binom{m+n-3}{m-1} = \binom{m+n-2}{m-1},$$

$$\text{it follows that } R(m-1,n) + R(m,n-1) \leq \binom{m+n-2}{m-1} \quad (*)$$

Now let $p = R(m-1,n) + R(m,n-1)$ and suppose that each edge of K_p is arbitrarily colored red or blue, analogous to exists or does not exist. We show that there is either a red K_m or a blue K_n . Let v be a vertex of K_p . Then the degree of $v = p - 1 = R(m-1,n) + R(m,n-1) - 1$. We consider two cases:

Case 1. Assume that v is incident with at least $R(m-1,n)$ red edges. Let S denote the set of vertices of K_p that are joined to v by red edges. Thus $|S| \geq R(m-1,n)$ and (S) is a complete graph of order at least $R(m-1,n)$ whose edges are colored red or blue. Therefore, (S) contains either a red K_{m-1} or a blue K_n . If (S) contains a blue K_n , so does K_p . Suppose that (S) contains a red K_{m-1} . Then $(S \cup \{v\})$ contains a red K_m . Hence, in this case, K_p contains either a red K_m or a blue K_n .

Case 2. Assume that v is incident with at most $R(m-1,n) - 1$ red edges. Then v is incident with at least $R(m,n-1)$ blue edges. Let T denote the set of vertices of K_p that are joined to v by blue edges. Therefore, $|T| \geq R(m,n-1)$ and (T) is a complete graph of order at least $R(m,n-1)$ whose edges are colored red or blue. Hence, (T) contains either a red K_m or a blue K_{n-1} . If (T) contains a red K_m , then K_p does as well. Suppose that (T) contains a blue K_{n-1} . Then $(T \cup \{v\})$ contains a blue K_n . In this case as well, then K_p contains a red K_m or a blue K_n .

This shows that $R(m,n) \leq R(m-1,n) + R(m,n-1)$, which when combined with (*), gives the desired result. \square

For our case, this theorem yields the result that $R(5,5) \leq 70$. This of course is far more than the current upper bound, which is 49. The Corollary to Theorem 1, however gives us a little closer approximation to the current upper bound. [5] The Corollary to Theorem 1 can also be thought of as a recursive definition of Theorem 1.

Corollary to Theorem 1: $R(m,n) \leq R(m-1,n) + R(m,n-1)$; if $R(m-1,n)$ and $R(m,n-1)$ are both even, then the inequality is strict.

Result: $R(m,n) = R(n,m)$

The previous corollary and result used together yield a new upper bound to $R(5,5)$ as long as we know $R(4,5)$, which is known to be 25. [7] Therefore, we have a new upper bound to $R(5,5)$, 50. Therefore, after another theorem is used, $R(5,5) \leq 49$. Our goal of this project is to improve the lower bound of $R(5,5)$ from 43 to 44.

2.2.2. Examples of Similar Ramsey Numbers

R(3,3):

$R(3,3)$ is similar to $R(5,5)$ only it is a bit smaller and is already known to be 6. The proof of $R(3,3) = 6$ is given by Greenwood and Gleason in 1955. [7] A brief overview of this proof is shown when looking at the bounds of $R(3,3)$. We know $R(3,3) > 3$ by default. We also know by Theorem 1 that $R(3,3) \leq 6$. Therefore we need to show that there are graphs of size 4 and 5 that do not have either a clique of size 3 or an independent set of size 3 in them. An easy counterexample of each is a cycle of size 4 and a cycle of size 5. This yields that $R(3,3) \leq 6$ and $R(3,3) > 5$, therefore $R(3,3) = 6$.

R(7,7):

$R(7,7)$ is also similar to $R(5,5)$ only it is a bit larger. The exact value of $R(7,7)$ is unknown, but it is known to be bounded by $205 \leq R(7,7) \leq 540$. This is much larger than the possible values of $R(5,5)$ and we only increased our m and n by 2. Therefore, it is clear that the general case of finding an exact answer to a Ramsey Number is far too difficult to do in a polynomial time algorithm (i.e. it is an NP-Complete Problem).

3. Approach to Finding Lower Bounds

The lower bound of $R(5,5)$ has been improved by the process of presenting counterexamples (i.e. a graph of size r such that no clique or independent set of size 5 exist in that graph). The most recent transition of the lower bound (from 42 to 43) came about through the use of Genetic Algorithms to search some of the graphs of size 42 for a counterexample. [4] This is a similar approach to the one we used in our project. There

are many different ways in which to go about this, all of which come down to finding a counter example that disproves the fact that 43 is the lower bound.

3.1. Enumeration of all Possibilities

The first and most obvious approach is to enumerate all possible graphs of size 43 and check if all of these graphs have a clique or independent set of size 5. Before beginning to write out all possible graphs of size 43, however, let's first look at how many possible graphs of size 43 exist.

There are 903 possible edges in a graph with 43 vertices and each possible edge has two possibilities, either an edge exists or it doesn't. Therefore there are 2^{903} possible graphs of size 43. This works out to be about 6.76×10^{271} possible graphs, which are far too many graphs to enumerate. Therefore this is not a feasible method.

3.2. Random Search

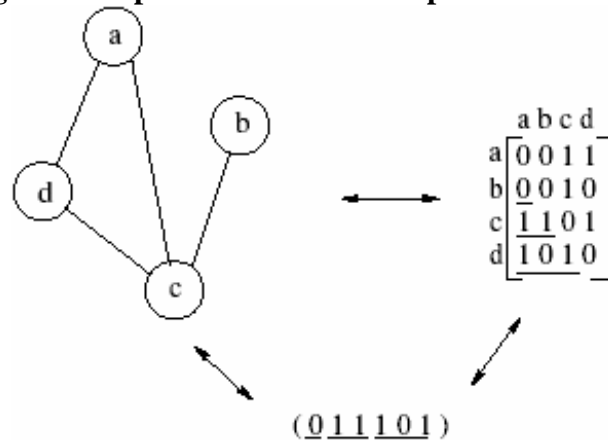
The next logical progression is to randomly search the entire search space and hope to get lucky. You have a better chance of winning the lottery than finding a counterexample using this method. Therefore, this approach is also infeasible.

3.3. Genetic Algorithms

The method we are going to use to search for a counterexample is Genetic Algorithms (GA). Genetic algorithms are a population based search algorithm based loosely on concepts from biological evolution. This section illustrates this process by using our problem of searching for an $R(5, 5)$ counterexample in the following sections:

3.3.1. Representation

Since the key data structure used in Genetic Algorithms is bit strings, we must have a way to represent a graph as a bit string. To do this, the graph's incidence matrix is created. An incidence matrix, M , of a graph with 43 vertices is a 43×43 matrix where each row, i , and column, j , represent a vertex of the graph and the edge that connects vertex i to vertex j is represented by position $M_{i,j}$ in the matrix. In this case, a 1 represents the existence of an edge, whereas a 0 represents the lack of an edge. See Figure 2 for an example.

Figure 2: Representation of a Graph as a Bit String

3.3.2. Initialization

The next step is to select a random sample of individuals from the search space. In this case, the search space is all the different graphs of size 43 (approximately 6.76×10^{271}). From this, a sample of 250 individuals was selected. The random samples are independent from run to run. This initialization is analogous to the random search technique discussed in Section 3.2.

3.3.3. Evaluation of Fitness

Now that we have 250 individuals, we must conduct tournaments to see which graphs are better than others. In order to conduct the tournaments, we must have a way of ranking or ordering the individuals. This is called the fitness function. Upon creation of our fitness function, examination of current maximum clique algorithms and approximation algorithms was very helpful [2]. In our case, the fitness of an individual is the number of cliques of size 5 in the graph plus the number of independent sets of size 5 in the graph. This was chosen because of the direct application to the problem. The possible fitness values for an individual from this fitness function range from 0 to 962598 (taken from K_{43}). A graph with fitness value of 0 would have no cliques of size 5 and no independent sets of size 5 and thus would be the desired counterexample.

3.3.4. Tournament

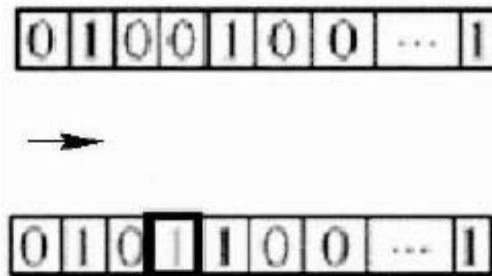
Now that each individual has a fitness value, tournaments can be conducted in order to rank our population. Using a tournament size of 2, individuals compete for chances to produce offspring. In our case, 2 individuals are chosen at random from the population. These two individual's fitness values are compared and the one with the better (i.e. smaller) fitness value is chosen to be a parent. This can be related to the "survival of the fittest" idea in biological evolution.

3.3.5. Recombination

The better individuals are now chosen to produce offspring through point mutation and one-point crossover. We chose to have a 50% Mutation rate, meaning that mutation takes place at approximately the same frequency as one point crossover.

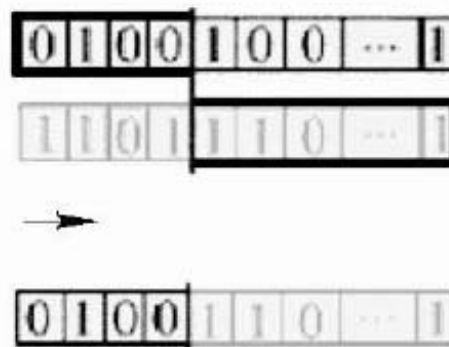
Point mutation involves randomly choosing one of our 903 bits in our bit string and flipping it, i.e. making it a 0 if it was a 1 and vice versa. This translates to the removal or addition of an edge to the graph, thus creating a new graph. An example of mutation is given in Figure 3.

Figure 3: Mutation Recombination Technique



One-point crossover involves taking two individuals and merging them together. To do this, a random point, n , must be chosen such that the new individual consists of bits 0 to $n-1$ from the first parent and bits n to 902 from the second parent. An example of one point crossover is given in Figure 4.

Figure 4: One-point Crossover Recombination Technique



3.3.6. Replacement

Now that new offspring are created, they replace the poorer individuals in the population as was decided by the tournaments. In the standard replacement, all of the new offspring are replaced at the same time at the end of one generation. In our case, however, we are using a steady state system with 25% replacement. This means that one individual is replaced every time a tournament completes. After 25% of the population (63 individuals) is replaced, one generation is complete.

3.3.7. Repeat

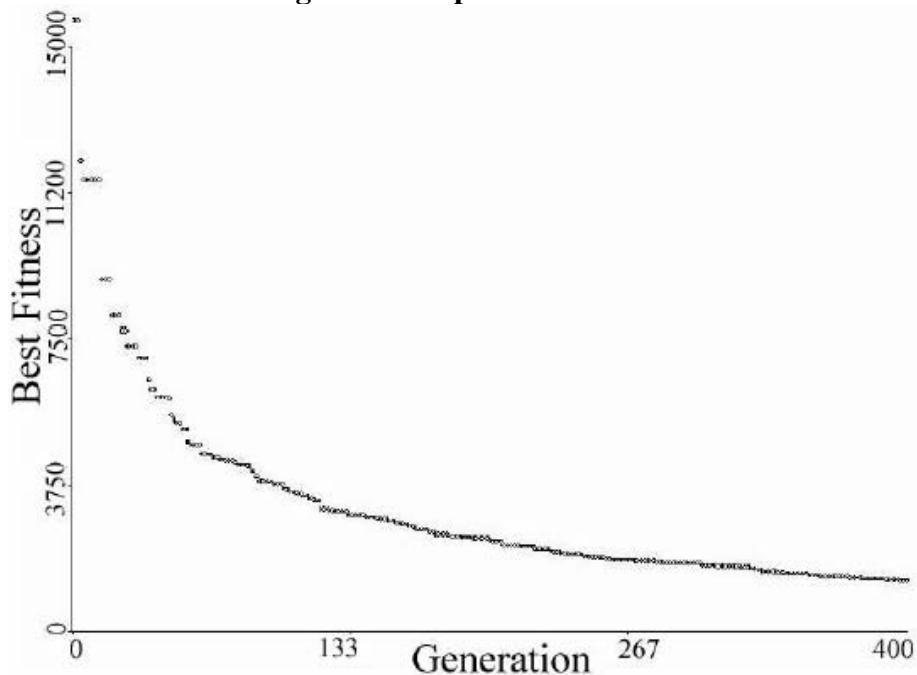
Evaluation of Fitness, Tournament, Recombination, and Replacement are repeated for as many generations as is chosen by the user. In our case 400 generations was chosen because after about 400 generations there was little improvement in the best fitness. This is due to the decreased variation in the population, meaning that all of the individuals began to look the same or strikingly similar.

For more information on Genetic Algorithms, see [1].

4. Results and Analysis

Over a period of 2 weeks, the Genetic Algorithm method described above was continuously run on 15 different machines in the Computer Science Software Development Lab at the University of Minnesota, Morris (Sci 2610). After the two-week period, there were 230 complete runs where data could be extracted and analysis of the run could take place. Each run lasted approximately 3 days. We explored a total of over 5,000,000 graphs of the 6.76×10^{271} (0.000 % of the search space). Of these 230 different runs, 10 had an overall best fitness between 1,300 and 1,400 while the others were larger than 1,400 but smaller than 2,000. This is good compared to the average best fitness for random sampling of individuals, which is approximately 15,000. The specifications of the computers used are given in Appendix B. Figure 5 is a graph of the best fitness of the best run with generation on the x-axis and best fitness on the y-axis.

Figure 5: Output of Best Run



The following tables give an average of the values from the runs taken at the given generations and a percentile breakdown of the entire selection of graphs that were examined.

Table 1: Average Fitness at Different Generations

Generation	Average Best Fitness
0	15309
50	6428
100	4194
150	3105
200	2510
250	2130
300	1882
350	1710
400	1585

Table 2: % of Graphs in Given Fitness Ranges

Fitness Range	% of Graphs in Range
0 → 1500	2.3 %
1501 → 2000	28.9 %
2001 → 3000	29.8 %
3001 → 4000	12.5 %
4001 → 5000	7.2 %
5001 → ∞	19.3 %

All runs have a curve similar to the one shown in Figure 5. One property of Genetic Algorithms is that the variation of the fitnesses of the population converges to 0. This is why it is uncertain whether or not extending this graph to an infinite number of generations will improve the fitness to 0 or if it will plateau asymptotically at a larger value. The best graph we found is given by the chromosome in Figure 6 (parsing via the method mentioned in Section 3.3.1. will yield the appropriate graph):

Figure 6: Chromosome of Best Individual

```
0110011100000111001000011101100000000101001100100101111101110101001111111010001100100001001100111100101011
111000001010011101101110100101011001100110001101101000000011000010110100011111001110100010101011001110010001
1100001111010001010101110010000010111011101101011010000011000110001101001110110110111001011001101011110100
011111011010001100010100010100101011110101100010001100101111011011000010110000101001010101000101110110111
1100111011000010001000111101110100101010111101010111001101000111100011011101001101100100101100110000000100
11001011111100010010001011010110011010101001110010100111001001011100100100100100010110011110000100101010111
110101101000001111001111101100011111001010101000010011001110100100011100101011000011100010101110011101111000
101000110001010100100111100101111011010100001100010010101011000100010101110101010111101001110000110110101001
000010011110111001100101111011100001010
```

5. Conclusion

It is clear that this project did not improve the lower bound of $R(5,5)$. However, the genetic algorithm method does yield interesting results and therefore might in fact produce a counter example given further refinements. The method we implemented could be improved by changing the number of generations, the population size, the recombination techniques, or any other of many available “knobs” that can be tweaked.

It is clear that for a random search technique, the best fitness will be quite large (as is evident from Table 1 – Generation 0). However, through the process of Genetic Algorithms the best fitness decreased from an average of approximately 15,000 to 1,500, a very substantial improvement. This improvement shows that if $R(5,5) > 43$ the Genetic Algorithm method is a reasonable technique to find a counterexample. With other techniques and advanced technology, the future is looking bright for finding a solution to this problem, or at the very least improving the lower bound of $R(5,5)$ to 44.

6. Future Research Work

Future work in this area could include many different changes in the possible “knobs” that can be tweaked in our Genetic Algorithm method. These include changing the number of generations, the population size, the mutation rate, the recombination techniques, the fitness function, or any of countless others. I experimented a bit with changing the population size and number of generations, but more work on that area would definitely prove to be useful in the future. Other possible research areas related to this include improving the algorithm used in computing the fitness of an individual and possibly using a different searching technique such as hill climbing. One could also take the best individuals found using my technique and compile all of them into an initial population for another Genetic Algorithm run. These are but a few of the many possibilities for future research in this area.

7. Acknowledgements

I would like to thank the Undergraduate Research Opportunities Program for partially supporting this project. I would also like to thank Prof. Peh Ng and Prof. Nic McPhee for their help advising me throughout this long process. I would like to thank Prof. Mark Logan for being a second reader for this paper. Finally, I would like to thank Valerie Cassel for all of her continued support and help throughout the process.

8. References

- [1] Banzhaf, Nordin, Keller, Francone, *Genetic Programming~ An Introduction*, Morgan Kaufmann Publishers, Inc., San Francisco, CA (1998).
- [2] Bomze, Budinich, Pardalos, Pelillo, *The Maximum Clique Problem*, (1999).
- [3] Bondy & Murty, *Graph Theory With Applications*, The Macmillan Press Ltd, Great Britain (1976).
- [4] Exoo, G., *A Lower Bound for $R(5,5)$* , Journal of Graph Theory, (1989).
- [5] Graham, Rothschild, Spencer, *Ramsey Theory*, John Wiley & Sons, New York, NY (1990).
- [6] Micron PC, <http://support.buympc.com/index.html>, 2003.
- [7] Radziszowski, Stanislaw P., *Small Ramsey Numbers (Revision #9)*, (2002).
- [8] Weisstein, Eric. *Eric Weisstein's World of Mathematics: A Wolfram Web Resource*, <http://mathworld.wolfram.com/> (2002).
- [9] West, Douglas B., *Introduction to Graph Theory*, Prentice-Hall, Inc., Upper Saddle River, NJ (1996).

9. Appendices

Appendix A: Number of Cliques of Size 5 Algorithm

```

/**
 * This method is an exact algorithm that returns the number of
 * cliques of size 5 in this graph.
 */
public int numCliquesSize5(Graph g) {
    int[] temp = new int[size];

    for(int i = 0 ; i < g.getSize() ; i++) {
        temp[i] = g.getDegree(i);
    }

    Graph tempG = g;

    int[] temp2 = new int[0];

    for(int i = 0 ; i < g.getSize() ; i++) {
        if(temp[i] >= 4) {
            temp2 = add(temp2,i);
        } else {
            tempG = remove(tempG,i);
        }
    }

    int out = 0;

    boolean[] intersect = new boolean[tempG.getSize()];
    intersect = initAll(intersect);
    boolean[] intersect1 = new boolean[tempG.getSize()];
    boolean[] intersect2 = new boolean[tempG.getSize()];
    boolean[] intersect3 = new boolean[tempG.getSize()];
    boolean[] intersect4 = new boolean[tempG.getSize()];
    boolean[] intersect5 = new boolean[tempG.getSize()];

    for(int x1 = 0 ; x1 < (temp2.length - 4) ; x1++) {
        intersect1 = intersection(intersect, connectedList(tempG, temp2[x1]));

        for(int x2 = (x1 + 1) ; x2 < (temp2.length - 3) ; x2++) {
            intersect2 = intersection(intersect1, connectedList(tempG, temp2[x2]));
            for(int x3 = (x2 + 1) ; x3 < (temp2.length - 2) ; x3++) {
                intersect3 = intersection(intersect2, connectedList(tempG, temp2[x3]));
                for(int x4 = (x3 + 1) ; x4 < (temp2.length - 1) ; x4++) {
                    intersect4 = intersection(intersect3, connectedList(tempG, temp2[x4]));
                    for(int x5 = (x4 + 1) ; x5 < temp2.length ; x5++) {
                        intersect5 = intersection(intersect4, connectedList(tempG, temp2[x5]));

                        if(numTrue(intersect5) > 4) {
                            out++;
                        }
                    }
                }
            }
        }
    }

    return out;
}

```

Appendix B: Computer Specifications

The following table is a list of specifications for the computers that were used for the Genetic Algorithm method. The table was taken from Micron's web site, the manufacturer of the machines used. [6]

Category	Description	QTY	Part Number
Processor	Intel Pentium III 1.6GHz 400FSB 256K 478Pin	1	<u>CPU002049-01</u>
Memory	***512MB 64X64 133MHZ SDRAM	1	<u>MOD001966-00</u>
Hard Drive	Seagate U6 20GB IDE 5400RPM	1	<u>HDI001793-00</u>
CD-ROM Drive	Lite-On 52X IDE CD-ROM LTN526	1	<u>CDI001263-01</u>
Video Card	Visiontek Vanta 16MB Video Card	1	<u>VCD001472-01</u>
Misc I/O	Microsoft 104-Key Elite Natural Keyboard	1	<u>KBR001062-03</u>
Misc I/O	Microsoft Intellimouse Optical USB and PS/2 Compatible	1	<u>MOU001072-01</u>
Case	Odyssey Client Pro PA700 front foot	1	<u>CSE001587-00</u>
Case	Odyssey ATX PA700 MiniTower w/o Power Supply	1	<u>CSE001638-00</u>
Power Supply	Delta 300W Power Supply	1	<u>PWS001108-00</u>
Operating System	Microsoft Windows 98 SE Recovery Media Kit	1	<u>OSS001318-00</u>
Software	Havre - Driver CD	1	<u>SFD001106-02</u>
Software	Microsoft Intellimouse Driver 3.5 in. Floppy (English)	1	<u>SFO001446-07</u>
Software	Microsoft.Internet Explorer 5.5	1	<u>SFO002447-00</u>
Warranty	ASSY DESKTOP THREE-YEAR LIMITED PARTS WARRANTY AND TECHNICAL SUPPORT POLICY	1	<u>WAR001049-00</u>
Case	Odyssey PA700 ATX MT Bezel Assembly	1	<u>BZL001057-03</u>
Motherboard	Havre - Intel 845 Brookdale Motherboard (Millennia® Max XR and ClientPro® CR)	1	<u>MBD001151-03</u>
Software	Nvidia Video Driver CD	1	<u>MED001417-05</u>