2006

The Study of Ramsey numbers r(C_k, C_k, C_k)

Yan Li

Rochester Institute of Technology
Department of Computer Science

Master's Project Report

The Study of Ramsey Numbers $r(C_k, C_k, C_k)$

Yan Li

January 2004

Committee:

Chairman: Prof. Stanisław P. Radziszowski

Reader: Prof. Peter G. Anderson

Observer: Prof. Ankur M. Teredesai

Abstract

The Ramsey number $r(C_k, C_k, C_k)$, denoted as $r_3(C_k)$, is the smallest positive integer n such that any edge coloring with three colors of the complete graph on n vertices must contain at least one monochromatic cycle C_k. In this project, most literature on the Ramsey numbers $r_3(C_k)$ are overviewed. Algorithms to check if a graph G contains any specific path or cycle and to construct extremal graphs for cycle C_k are developed. All good 3-colorings of complete graph K_{10} are constructed to verify the value of Ramsey number $r_3(C_4)$. Ramsey number value of $r_3(C_3)$ is verified by direct point by point extension algorithm. The lower bounds for the Ramsey numbers $r_3(C_5)$, $r_3(C_6)$, and $r_3(C_7)$ are provided as well. Additionally, the possibility of further research for larger k, especially for $r_3(C_8)$ and $r_3(C_{10})$ is investigated. Most of the results are based on computer algorithms.

URL: http://www.cis.rit.edu/ yxl4059/ramsey.html

Acknowledgments

I would like to thank my advisor, Prof. Stanisław P. Radziszowski, for his generous advice, guidance and inspiration from the start to the very end. He had a well defined task for me and he also worked on improving my English. It has been a pleasure to have Prof. Stanisław P. Radziszowski as my advisor. I would also like to thank Dr. Peter G. Anderson for his reading of my report and his insightful comments. Thanks also go to Prof. Ankur M. Teredesai for his being there of my defense.

Contents

Chapter 1

Introduction

In this project, what has been studied are the Ramsey numbers $r(C_k, C_k, C_k)$, which is the smallest positive integer n such that any edge coloring with three colors of the complete graph on n vertices must contain at least one monochromatic cycle of length k. Definitions and notation in graph theory are given in Chapter 2. An overview of most literature on the Ramsey numbers $r(C_k, C_k, C_k)$ is also given in Chapter 2.

In Chapter 3, approaches are presented to check if a graph contains any specific path or cycle and to construct extremal graphs for a cycle on k vertices. Also, two different algorithms for verifying the Ramsey number values of $r(C_3, C_3, C_3)$ and $r(C_4, C_4, C_4)$ based on the idea that was first established by Bialostocki [1] and Clapham [4] in 1984 are presented. Another algorithm is also given in Chapter 3 to obtain good lower bounds for Ramsey numbers $r(C_5, C_5, C_5)$, $r(C_6, C_6, C_6)$, $r(C_7, C_7, C_7)$, $r(C_8, C_8, C_8)$ and $r(C_{10}, C_{10}, C_{10})$.

In Chapter 4, the software tools that we utilized and implemented are discussed. The detailed description of results and some final thoughts are given in Chapter 5 and Chapter 6, respectively.

Chapter 2

Theoretical Background

2.1 Definions and Notation

A *graph*, $G = (V(G), E(G))$, is a set of vertices and set of unordered pairs of distinct elements of vertices, where $V(G)$ is vertex set and $E(G)$ is edge set. A *multigraph* is a graph that contains a pair of vertices connected by multiple edges. A *loop* is an edge that connects a vertex to itself. If edges are given direction, the graph is called a *directed graph* or *digraph*. Suppose (u, v) is an edge in a graph or digraph, then vertex u is *adjacent* to vertex v. In this project, we only consider undirected finite graph without any loops and multiple edges.

There is a matrix naturally associated with a graph called *adjacency matrix*. The adjacency matrix $A = A(G) = [a_{ij}]$ of graph G is a $n \times n$ matrix given by

$$a_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

The matrix A is symmetric and its trace is zero.

A *path* P_k defined on k vertices $V(P_k) = \{v_1, v_2, \ldots, v_k\}$ is the set of edges $E(P_k) = \{v_1v_2, v_2v_3, \ldots, v_{k-1}v_k\}$. The length of a path is the number of edges traversed, equivalently it is equal to $k - 1$. A *cycle* C_k defined on k vertices $V(C_k) = \{v_1, v_2, \ldots, v_k\}$ is the set of edges $E(C_k) =$

$\{v_1v_2, v_2v_3, \ldots, v_{k-1}v_k, v_kv_1\}$. A path or cycle is a *simple path* or *simple cycle* if it has no repeated vertices. A *complete graph* K_n is a graph on n vertices with edges connecting any pair of vertices. A graph G_1 is a *subgraph* of a graph G_2 if G_1 is isomorphic (defined below) to a graph all of whose vertices and edges are in G_2.

In graph theory, two graphs are *isomorphic* if they can be represented by identical diagrams or adjacency relationship. We can make this idea exact by defining the notation of graph isomorphism: a graph G is isomorphic to H iff there exists a one-to-one mapping $\phi : V(G) \to V(H)$ that preserves adjacency, namely, $\{u, v\} \in E(G)$ if and only if $\{\phi(u), \phi(v)\} \in E(H)$.

The *extremal graph number* $ext(H, n)$ denotes the maximal number of edges of a graph with n vertices, which does not contain a subgraph isomorphic to H [7]. In the project we only consider H as a cycle C_k. For example, if we consider H as cycle C_4, we have asymptotically $ext(H, n) \sim \frac{1}{2}n^{\frac{3}{2}}$ [2].

The classical problem in Ramsey theory is the party problem which asks the minimum number of guests $r(m, n)$ that must be invited so there are at least m people will know each other or at least n people won't know each other. The solution $r(m, n)$ is called *Ramsey number*. To get a typical result in Ramsey theory, we consider partitions of the edges of graphs and a partition will be called *edge coloring*. An *edge coloring* $(r_1, r_2, \ldots, r_k)$, $r_i \geq 1$ for $1 \leq i \leq k$, is an assignment of one of k colors to each edge of a complete graph G, such that it does not contain monochromatic complete subgraph K_{r_i} in color i, for $1 \leq i \leq k$. By the definition of edge coloring, adjacent edges may have the same color. To verify the Ramsey number, our aim is to show there exist large subgraphs all of whose edges with the same color.

The *classical Ramsey number* $r(r_1, r_2, \ldots, r_k)$ is defined to be the least integer $n > 0$ such that for a complete graph K_n with any edge coloring method $(r_1, r_2, \ldots, r_k)$, there always exists at least one monochromatic complete subgraph K_{r_i} in color i for $1 \leq i \leq k$. Hence, we can color the edges of K_n in some fixed number k of colors and then find some minimal number of vertices so that any edge coloring in k colors forces a monochromatic subgraph that is isomorphic to K_{r_i}. Very few Ramsey numbers are known

and improving upper bounds on Ramsey numbers is a very hard problem. In this project, we concentrate on a special case $r_3(C_k) \equiv r(C_k, C_k, C_k)$, which is the smallest positive integer n such that any edge coloring with three colors of the complete graph on n vertices must contain at least one monochromatic cycle of length k. Since the goodness of an edge coloring is related to three parameters k, n, and c, where k is the number of vertices of a cycle, n is the number of vertices of a complete graph, and c is the number of colors, therefore, we can call an edge coloring of K_n k-good if there is no monochromatic cycle C_k formed. Therefore, the Ramsey number $r_3(C_k, C_k, C_k)$ is the smallest positive integer n for which there is no k-good 3-coloring of K_n.

2.2 Historical Overview

A good and detailed overview of known bounds and exact values of various types of Ramsey numbers is given in Radziszowski's survey [15]. The classical Ramsey numbers with 2 colors have been studied thoroughly and many results have been obtained. However, in the multicolor case (number of colors $c > 2$), it becomes more complicated to find general results or to determine exact values of Ramsey numbers. In this case the only known nontrivial value of the classical Ramsey number for avoiding complete graphs is $r_3(3) = r_3(C_3) = r_3(K_3) = 17$. Up to now for the special case of Ramsey numbers of the form $r_3(C_k)$, there are only five known exact values shown in Table 2.1. Obviously, $r_3(C_3)$ and $r_3(K_3)$ are the same case.

Ramsey number	value
$r_3(K_3) = r_3(C_3)$	17
$r_3(C_4)$	11
$r_3(C_5)$	17
$r_3(C_6)$	12
$r_3(C_7)$	25

Table 2.1: Values of Ramsey numbers of form $r_3(C_k)$

In a 1955 article by R. E. Greenwood and A. M. Gleason, the authors showed $r_2(3) = 6$ and gave the first proof of $r_3(3) = 17$. From $r(3,3) = 6$, we know that, for example, if we color the edges of a complete graph on six or more vertices with red and blue then there must be a red or blue triangle. On the other hand, we can find a red-blue coloring of the edges of K_5 that avoids monochromatic triangles by coloring a pentagon in red and inscribing a star of blue within. Obviously, this is a 3-good 2-coloring of K_5. Greenwood and Gleason also showed that if K_n has a good 2-coloring then $n \leq 5$. Based on $r_3(3) = 17$ we know that if we color the edges of a complete graph in three colors then no matter what coloring we use we must have a monochromatic triangle if the complete graph has seventeen or more vertices. The proof of $r_3(3) = 17$ includes two parts. First, Greenwood and Gleason gave a 3-good 3-colorings of K_{16} to show $r_3(3) \geq 17$. Then the authors proved that any 3-coloring of K_{17} must contain a monochromatic triangle.

In 1984, A. Bialostocki and J. Schönheim established $r_3(C_4) = 11$ by giving a coloring of the edges of K_{10} with 3 colors with no monochromatic C_4 to show $r_3(C_4) \geq 11$; on the other hand, they proved the upper bound of $r_3(C_4)$ to be 11 [1]. They proved this upper bound by establishing $ext(C_4, 11) \geq 18$ due to $E(K_{11}) = 55 > 3 \times 18$, which is a consequence of $ext(C_4, 10) = 16$ because of $E(K_{10}) = 45 \leq 3 \times 16$. The authors proved $ext(C_4, 10) = 16$ by the following two propositions:

Proposition 1 If a graph G with 10 vertices not containing any C_4 as a subgraph has more than 15 edges, then the maximal degree Δ of its vertices is 4.

Proposition 2 If a graph with 10 vertices not containing any C_4 as a subgraph has maximal degree $\Delta = 4$, then
(i) It has at most 16 edges;
(ii) There are only two extremal graphs.

Bialostocki and Schönheim also showed a partition of edges of K_{10} into three C_4-free classes shown in Figure 2.1. The three graphs H_1, H_2 and H_3 are not isomorphic. They proved such a partition is impossible for K_{11} since

$ext(11, C_4) = 18$.

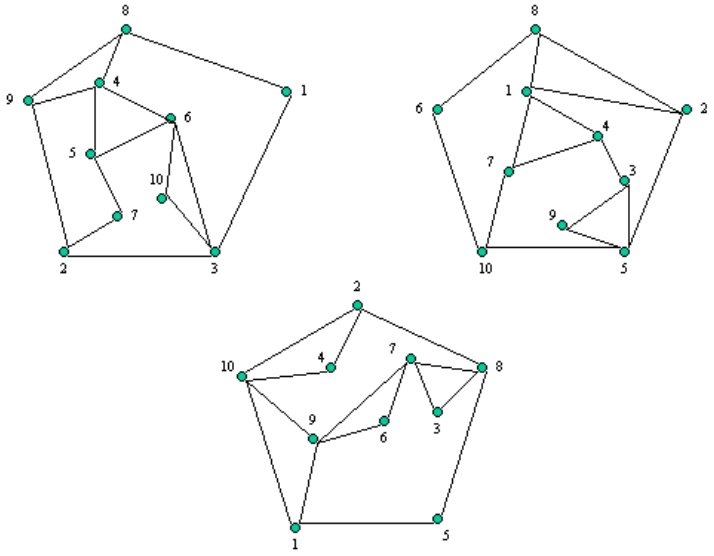

Figure 2.1: H_1, H_2 and H_3

In 1987 C. R. J. Clapham found a coloring with isomorphic parts for the edges of K_{10} that shows $r_3(C_4) \geq 11$. In the paper, he exhibited one C_4-free graph formed by the red edges. Then he colored the remaining edges blue and green by using permutation of the vertices to induce a permutation of the edges taking red edges to blue, blue edges to green, and green edges to red [4]. The graphs formed by the blue and green edges are isomorphic to the graph formed by the red edges.

In 1992, Y. Yang and P. Rowlinson showed that the Ramsey numbers $r_3(C_5)$ and $r_3(C_6)$ are equal to 17 and 12, respectively, in their papers [16] and [18]. They obtained the values of $ext(C_6, n)$, $6 \leq n \leq 21$, and all of the corresponding extremal graphs by computer [18]. At the same time, sufficient information was obtained to show $r_3(C_6) = 12$.

In 2001, R. Faudree, A. Schelten and I. Schiermeyer proved that $r_3(C_7) = 25$ without using relying on computer algorithms [7].

We summarize the overview of history of Ramsey numbers $r_3(C_k)$ in Table 2.2.

reference	year	result
Greenwood and Gleason [10]	1955	$r_3(C_3) = r_3(K_3) = 17$
Bialostocki and Schönheim [1]	1984	$r_3(C_4) = 11$
Clapham [4]	1987	$r_3(C_4) = 11$
Yang and Rowlinson [16]	1992	$r_3(C_5) = 17$
Yang and Rowlinson [18]	1993	$r_3(C_6) = 12$
Faudree, Schelten and Schiermeyer [7]	2003	$r_3(C_7) = 25$

Table 2.2: Historical Overview

Chapter 3

Approaches and Algorithms

3.1 Methods of checking the existence of path and cycle

In this project, all algorithms dealing with multiple colors were designed for specific case of 3 colors. To verify the values of Ramsey number $r_3(C_k)$, we need to be able to find out quickly if the edge coloring of K_n is k-good or not. Hence, methods that can be applied to check the existence of specified monochromatic cycle are needed. In this section, a method called *path search method* to check if a graph G contains any specific monochromatic path is presented. According to this method, the existence of any monochromatic cycle C_k is obtained by *cycle search method*. In the following, the underlying thought, pseudo-code and some significant function definitions of each method are given step by step.

3.1.1 Path search method (PS)

The PS method checks if a graph G contains any monochromatic simple path P_k by eliminating one vertex and checking if there exists a path of length $k-1$ recursively.

PS(k, G)

input: k, integer > 1, number of vertices of path; G, graph.
output: boolean. 1 if a path of k vertices found in graph G; 0 if no path of k vertices exists in graph G.

procedure:

```
PS(k,G)
n <- NUMVER(G)
for i of 1, n - 1, 1 do
  for j of i + 1, n, 1 do
    if PATH(i, j, k, G) then
      return 1
    endif
  endfor
endfor
return 0
```

- PATH(v_1, v_2, k, G)
  input: v_1, a vertex in graph G; v_2, another vertex in graph G; k, integer > 1, number of vertices of path.
  output: boolean. 1 if a path of k vertices found between vertices v_1 and v_2; 0 if no path of k vertices exists between vertices v_1 and v_2 .

  procedure:

```
PATH(v1, v2, k, G)
n <- NUMVER(G)
if n <= 1 or k <= 1 then
  return 0
endif
if k = 2 and EDGE(v1, v2, G) = 1 then
  return 1
else
```

```
  return 0
endif
a[1, 2 ..., m] <- ADJACENT(v1, G)
for i of 1, m - 1, 1 do
  if a[i] != v2 then
    r = PATH(a[i], v2, k - 1, SUB(v1, G))
    if r = 1 then
      return 1
    endif
  else
    return 1
endfor
return 0
```

- NUMVER(G)
  input: G, graph.
  output: n, number of vertices of graph G.
  procedure: obtain the number of verrices of graph G.

- EDGE(v_1, v_2, G)
  input: v_1, v_2, two vertices in graph G; G, graph.
  output: boolean. 1 if there exists an edge between vertices v_1 and v_2; 0 if no edge exists between vertices v_1 and v_2.
  procedure: check if there exists an edge between vertices v_1 and v_2 in graph G.

- ADJACENT(v_1, G)
  input: v_1, a vertex in graph G; G, graph.
  output: a vertiex set containing all adjacent vertices of v_1.
  procedure: obtain all adjacent vertices of vertex v_1 in graph G.

- SUB(v_1, G)
  input: v_1, a vertex in graph G; G, graph.
  output: a graph H on $n - 1$ vertices without vertex v_1.

procedure: eliminate the vertex v_1 from graph G and return the result graph.

3.1.2 Cycle search method (CS)

The CS method checks if there exists a simple cycle of k vertices in a graph G by checking if there is an edge between the starting and ending vertices of the path based on the previous results of path searching. It also calls function NUMVER, EDGE, and PATH.

CS(k, G)
input: k, integer > 1, number of vertices of cycle; G, graph.
output: boolean. 1 if a cycle of k vertices found in graph G; 0 if no cycle of k vertices exists in graph G.

procedure:

```
CS(k, G)
n <- NUMVER(G)
for i of 1, n - 1, 1 do
  for j of i + 1, n, 1 do
    if EDGE(i, j, G) = 1 then
      if PATH(i, j, k, G) = 1 then
        return 1
      endif
    endif
  endfor
endfor
return 0
```

3.2 Methods of constructing C_k-free graphs

The *C_k-free graph construction method* (FG) is applied to construct all non-isomorphic C_k-free graphs of n vertices and find the extremal graph numbers $ext(C_k, n)$ by generating all non-isomorphic graphs on n vertices and eliminating the graphs with cycle C_k based on the results of cycle searching. At the same time, another method called $ISOFG$ is utilized to obtain all C_k-free graphs of n vertices including isomorphism. The difference in implementation between FG and $ISOFG$ is that, for the former, all non-isomorphic graphs are created by utility program *makeg*; however, for the latter, all graphs including isomorphism are created by exhaustive searching for each vertex with specialized program *isog*. Some results about cycle-free extremal graphs are shown in Chapter 5.

FG(k, n)
input: k, number of vertices of cycle; n, number of vertices of graph.
output: a graph set containing all C_k-free graphs on n vertices.

procedure:

```
FG(k, n)
graph_set[1, ..., m] <- GEN_GRAPH(n)
for i of 1, m, 1, do
  if CS(k, graph_set[i]) = 1 then
    REMOVE(i, graph_set)
  endif
endfor
return graph_set
```

- GEN_GRAPH(n)
  input: n, number of vertices of graph.
  output: a graph set containing all graphs on n vertices.
  procedure: generate all graphs on n vertices.

- REMOVE(i, *graph_set*)
  input: i, graph index; *graph_set*, a graph set containing all graphs on n vertices.
  output: a graph set containing only C_k-free graph on n vertices.
  procedure: remove the ith graph with cycle C_k from the graph_set.

3.3 Algorithms

Based on the definition of Ramsey number, the most direct way of verifying the Ramsey number values of $r_3(C_k)$ is generating all the 3-colorings of K_n, removing the colorings with a monochromatic cycle C_k, and finding the first n with empty output. However, due to the large time and space complexity, this naive method is not feasible for even moderate n. In this section, we outline two algorithms to construct the C_k-free 3-colorings of the edges of K_n and another heuristic random walk algorithm to obtain the lower bounds on Ramsey numbers $r_3(C_k)$.

3.3.1 Direct extension algorithm (DE)

The *DE* algorithm is an iterative point by point algorithm which generates all C_k-free 3-colorings of K_n ($n = 3, 4, \ldots$) from the starting two vertices by performing exhaustive enumerations of possible extensions.

DE(n, k)
input: n, number of vertices of K_n; k, number of vertices of cycle.
output: all good 3-colorings of K_n.

procedure:

```
DE(n, k)
if n < 3 return NIL
if n = 3 return BUILDG3()
graph_set <- NIL
```

```
s[1, ..., m] <- DE(n-1, k)
for i of 1, m, 1 do
  g[1, ..., p] <- ADDV(s[i])
  for j of 1, p, 1 do
    if CS(k, g[j]) = 0 then
      graph_set <- ADDG(graph_set, g[j])
    endif
  ednfor
endfor
return graph_set
```

- BUILDG3()
  input: no input parameter.
  output: all C_k-free 3-colorings of edges of K_3.
  procedure: construct three C_k-free 3-colorings of edges of K_3.

- ADDV(*coloring c*)
  input: a good 3-coloring of K_n.
  output: a set of all 3-colorings of K_{n+1}.
  procedure: generate all 3-colorings of edges of K_{n+1} by performing exhaustive enumeration of possible extensions of c.

- ADDG(*coloring_set*, *coloring*)
  input: *coloring_set*, set of C_k-free 3-colorings of edges of K_n; *coloring*, a C_k-free 3-colorings of edges of K_n.
  output: new set of C_k-free 3-colorings of K_n.
  procedure: add a C_k-free 3-coloring of K_n to the input coloring set.

However, the exponentially increasing number of graphs makes it not attainable to verify the Ramsey numbers with cycle larger than C_6 even if the time complexity is linearly dependent on the input number of vertices of graph G.

3.3.2 Matching algorithm (MR)

The *MR* algorithm is utilized to generate all (C_k, C_k, C_k) colorings from the C_k-free graphs based on the results of Section 3.2. To compose all good 3-colorings of K_n and cycle C_k by *MR*, each time two graphs on n vertices not containing cycle C_k are overlapped as color 1 and color 2. To avoid omitting any k-good 3-coloring, we employ each graph from the graph set of n vertices without C_k with isomorphism as color 2 instead of applying non-isomorphic graph set. However, for color 1, only non-isomorphic graph set is utilized. Edge conflicts between color 1 and color 2 are checked simultaneously. If no edge conflict is obtained, the algorithm checks if there is a cycle C_k in the remaining part of K_n which is assigned color 3 by *CS*. Eventually, it's a good 3-coloring if there is no cycle C_k in color 3.

MR(n, k)
input: n, number of vertices of graph; k, number of vertices of cycle.
output: all good 3-colorings of K_n.

procedure:

```
e <- EXT(k, n)
c[3][1, ..., m] <- GENC(e, n)
graph_set <- NIL
for i of 1, m, 1 do
  s1[1, ..., x] <- ISOFG(k, c[1][i])
  s2[1, ..., y] <- FG(k, c[2][i])
  for j of 1, x, 1 do
    for l of 1, y, 1 do
    if !(CONFLICT(s1[j], s2[l])) then
      g <- MONO3(n, s1[j], s2[l])
      if !(CS(k, g)) then
        graph_set <- ADDG(r, MULTI(s1[j], s2[l],g))
      endif
```

```
    endif
  ednfor
endfor
return graph_set
```

- EXT(k, n)
  input: k, number of vertices of cycle; n, number of vertices of graph.
  output: extremal graph number.
  procedure: find the maximal number of edges of a graph on n vertices without a subgraph isomorphic to C_k.

- GENC(e, n)
  input: e, extremal graph number; n, number of vertices of graph.
  output: an $3 \times m$ array denoting all the possible distributions of three colors of a graph.
  procedure: generate a 2D array to store the color distribution.

- CONFLICT(g_1, g_2)
  input: g_1, g_2, two monochromatic C_k-free graphs on the same number of vertices.
  output: boolean. 1 if there is an edge conflict between two graphs; 0 if no edge conflict exists.
  procedure: check edge conflict between two graphs g_1 and g_2.

- MONO3(n, g_1, g_2)
  input: n, number of vertices of complete graph; g_1, g_2, two monochromatic C_k-free graphs of the same number of vertices n.
  output: a monochromatic graph of n vertices which can be matched with the third color.
  procedure: obtain a monochromatic graph of n vertices.

- MULTI(g_1, g_2, g_3)
  input: three monochromatic C_k-free graphs with the same number of vertices n.

output: a C_k-free 3-coloring of K_n.
procedure: overlap the three C_k-free monochromatic graphs to obtain a good 3-coloring of K_n.

3.3.3 Trajectory algorithm (TM)

The TM algorithm is a heuristic version of DE algorithm which is not exhaustive, but can at least establish lower bounds for Ramsey numbers. The idea is to choose a cycle-free 3-coloring of K_n randomly, generating more cycle-free 3-colorings of K_{n+1} recursively, and repeating the trajectory generating procedure as much as possible. A computing chain of multi-colorings with vertices $2, 3, 4, \ldots$ is regarded as a trajectory. More trajectories are attempted, closer to the result of DE algorithm can be obtained.

TM(k, n)
input: k, integer > 1, number of vertices of cycle; n, number of vertices of graph.
output: a C_k-free 3-coloring of K_n.

procedure:

```
TM(k, n)
if n < 3 return NIL
if n = 3 return BUILDG3()
while CS(k, g) = 0 do
  s <- randomly pick one graph from TM(k, n)
  g <- ADDV(s)
  g <- randomly pick one graph from TM(k, n+1)
enddo
return graph
```

The TM algorithm requires small hard disk space and has the same computational complexity as DE. However, at any time, it can only obtain the lower bound, instead of the accurate values of Ramsey numbers. When

more and more trajectories are attempted, with an increasing certainty, more accurate lower bounds on Ramsey numbers are obtained.

Chapter 4

Software Tools

4.1 General Utility Programs

There are two formats which were applied to represent the adjacency matrix of a given graph. Y-format is an $ASCII$ format in which the lower 6 bits of a byte are used to maintain the adjacencies. Each byte has the format 01xxxxxx, where x represents one bit of data. For the first byte, xxxxxx represents the number of vertices n; the other ceiling $(n(n-1)/12)$ bytes contain the upper triangle of the adjacency matrix in column major order. That is, the entries appear in the order $(0,1),(0,2),(1,2),(0,3),(1,3),(2,3),\ldots$. The bits are used in left to right order within each byte. Any unused bits on the end are set to zero. The maximum supported number of vertices is 63 [12]. The following is an example of a y-format string and its corresponding adjacency matrix of a graph on 10 vertices not containing C_4 as a subgraph.

```
The string of y-format: J@@@@@A_‘
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1
```

```
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0
0 1 1 1 1 1 1 0 0 0
```

Mc-format, developed by B. D. McKay and S. Radziszowski, used in [14], was applied to store and manipulate colorings of K_n, which allows each coloring to be represented by one line with two security bits in each byte. In mc-format, each byte has the value $63 + x$, where x represents six bits of data. Similarly as y-format, x is the number of vertices n for the first byte; for the second byte, x is the number of colors c. Let b be the minimum number of bits needed to hold $0 \dots c$. The other ceiling $(bn(n-1)/12)$ bytes contain the upper triangle of the adjacency matrix in column major order. The bits are used in left to right order within each byte, to make color values of unsigned b-bit integers. Any unused bits on the end are set to zero. Mc-format can handle the maximal numbers of vertices and colors up to 63 [12]. Consider a 4-good 3-coloring of K_{10} given in Chapter 5, identifying colors with 1, 2, and 3, the mc-format and corresponding adjacency matrix for this coloring are:

```
The string of mc-format: IB\h]ffxee}}|t}Xt
 132113233
1 12332123
31 1221232
221 212331
1322 31312
13213 2231
321212 313
2123323 11
32331311 1
332121311
```

The function *canlab* is applied to obtain the canonical labeling of graphs. For two graphs G_1 and G_2, they are isomorphic if and only if canlab(G_1) = canlab(G_2). In this project, program *labely* and *labelmc* developed by B. D. McKay and computing a canonical labeling of graphs and multicolorings, are applied [12]. Thus, we convert the isomorphism problem into identity which is solved by the UNIX system command sort -u, which deletes identical lines. *Shortmc*, also developed by B. McKay, is applied to remove isomorphic duplicates from a file in mc-format. *Shortmc* run on an input file in mc-format generates an outfile in the same format containing a subset of the original colorings, one from each isomorphism class, each with the canonical labeling. For the graphs with y-format, *shorty* is utilized to put graphs in canonical forms. Additionally, there are some other utility programs which were used in this project, and they are summarized as follows.

- *makeg* – generate all graphs in y-format on n vertices; usage: makeg n.
- *picky* – select graphs from a file according to their parameters; usage: picky -option input output .
- *listy* –convert graphs from y-format to a human-readable form; usage: listy infile outfile.
- *seey* – generate adjacency matrix from y-format; usage: seey infile outfile.
- *listmc* – write graphs in human-readable format; usage: listmc infile outfile.
- *dcount* – count graphs by degree sequence; usage: dcount -option infile.
- *ecount* – count graphs by edge sequence; usage: ecount -option infile.

For example, given any set of graphs in y-format, with the following command

```
labely big.y | sort -u > small.y
```

generates exactly one graph in each isomorphism class of big.y and writes it to small.y, furthermore the graphs in small.y will be labeled canonically. The following command produces file result.y containing exactly one canonically labeled isomorphism of each graph on 20 vertices and 95 edges which appeared in the generated graph set in y-format.

```
gengraph | picky -n20 -e95 | labely > result.y
```

4.2 Specialized Programs

C++ was chosen as the software development language for its apparent advantages of object-oriented features and ease of bit manipulation features. Eight classes are designed in this project as follows.

- $YFormat$ – handling the operations on a y-format string.
- $MCFormat$ – handling the operations on a mc-format string.
- $Combine$ – generating m combinatorial numbers from n given numbers $C(n, m)$.
- $Path$ – recording the path between each pair of vertices in graph.
- $Ramsey$ – generating all cycle-free 3-colorings on n vertices based on cycle-free monochromatic graphs with the same number of vertices.
- $MCGraph$ – handling the operations on a multi-coloring in mc-format.
- $Graph$ – handling the operations on an adjacency matrix from y-format string.
- $Trajectory$ – obtaining the lower bound of Ramsey numbers by attempting trajectories which is a naive implementation of random walking simulation.

Chapter 5

Computations

In this chapter, the description of detailed results of different algorithms are given. The main Ramsey number under consideration is $r_3(C_4)$. Results on other Ramsey numbers $r_3(C_k)$ are presented as well. Additionally, results on some extremal graphs are also given.

It is true that if we color the edges of a complete graph in three colors then no matter what coloring we use we must have a monochromatic cycle on four vertices if the complete graph has eleven or more vertices. Formally, $r_3(C_4) = 11$. In this project, results on extremal graphs, $ext(C_4, n)$, are generated by computer programs, which shows the upper bound of Ramsey number $r_3(C_4)$ is 11; on the other hand, to verify $r_3(C_4) \geq 11$, all good 3-colorings of K_n ($n \leq 11$) are given in this project by two different algorithms. Hence, eventually, $r_3(C_4) = 11$ is verified.

5.1 Results on Extremal Graphs $ext(C_k, n)$

We summarize all the non-isomorphic graphs on 10 vertices not containing any cycle C_4 as a subgraph generated by specialized program *extremal* based on the number of edges as follows. The total number of graphs is 5069.

```
1 graphs with 10 vertices and   0 edges
1 graphs with 10 vertices and   1 edges
```

```
    2 graphs with 10 vertices and   2 edges
    5 graphs with 10 vertices and   3 edges
   10 graphs with 10 vertices and   4 edges
   23 graphs with 10 vertices and   5 edges
   53 graphs with 10 vertices and   6 edges
  119 graphs with 10 vertices and   7 edges
  262 graphs with 10 vertices and   8 edges
  529 graphs with 10 vertices and   9 edges
  883 graphs with 10 vertices and  10 edges
 1156 graphs with 10 vertices and  11 edges
 1097 graphs with 10 vertices and  12 edges
  670 graphs with 10 vertices and  13 edges
  222 graphs with 10 vertices and  14 edges
   34 graphs with 10 vertices and  15 edges
    2 graphs with 10 vertices and  16 edges
 5069 graphs altogether
```

Therefore, we can see that there are only two extremal graphs with $ext(C_4, 10) = 16$ edges, denoted by G_1 and G_2, which are given in Figure 5.1. The maximal degree is 4 which is consistent with Bialostocki and Schönheim's result [1]. The adjacency matrices of graphs G_1 and G_2 are given as below. From the above summary, we conclude that all graphs on 10 vertices with the number of edges greater than 16 must contain cycle C_4. Hence, $ext(C_4, 10) = 16$ is verified. Due to $E(K_{11}) = 55$ and $\lceil \frac{55}{3} \rceil = 19$, on the other hand, $ext(C_4, 11) < 19$ due to $ext(C_4, 11) = 18$, therefore, the upper bound of $r_3(C_4)$ is proved by the results of extremal graph.

```
G1: Graph on 10 vertices without C4
0 1 1 0 0 0 0 0 1 0
1 0 0 0 1 0 0 0 0 1
1 0 0 1 0 0 0 0 1 0
0 0 1 0 0 1 0 0 0 1
0 1 0 0 0 0 1 0 0 1
```

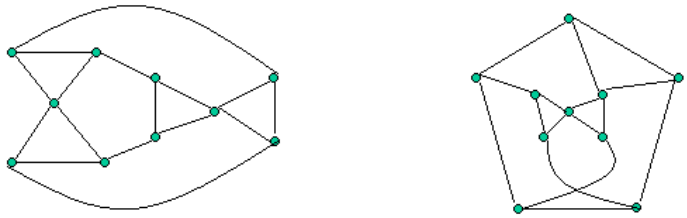

Figure 5.1: Two extremal graphs for $ext(C_4, 10)$

```
0 0 0 1 0 0 0 1 0 1
0 0 0 0 1 0 0 1 1 0
0 0 0 0 0 1 1 0 1 0
1 0 1 0 0 0 1 1 0 0
0 1 0 1 1 1 0 0 0 0
degrees are: 3 3 3 3 3 3 3 3 4 4

G2: Graph on 10 vertices without C4
0 1 1 0 0 0 0 1 0 0
1 0 0 1 1 0 0 0 0 0
1 0 0 0 0 1 1 0 0 0
0 1 0 0 0 1 0 0 1 0
0 1 0 0 0 0 1 0 0 1
0 0 1 1 0 0 0 0 1 0
0 0 1 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 1 1
0 0 0 1 0 1 0 1 0 1
0 0 0 0 1 0 1 1 1 0
degrees are: 3 3 3 3 3 3 3 3 4 4
```

More detailed results on $ext(C_4, n)$ are given in Table 5.1 and 5.2. Each entry in Tables 5.1 and 5.2 represents the number of graphs which have n vertices and the number of edges e. The extremal graph numbers, $ext(C_4, n)$, are given in Table 5.3. In Table 5.3, the last column shows the number of graphs if the number of edges is equal to the extremal graph number.

edge number e	0	1	2	3	4	5	6	7	8
2	1	1	0	0	0	0	0	0	0
3	1	1	1	1	0	0	0	0	0
4	1	1	2	3	1	0	0	0	0
5	1	1	2	4	5	4	1	0	0
6	1	1	2	5	8	12	11	4	0
7	1	1	2	5	9	18	29	30	17
8	1	1	2	5	10	21	43	72	88
9	1	1	2	5	10	22	50	103	188
10	1	1	2	5	10	23	53	119	262
11	0	0	0	0	0	0	0	0	0

Table 5.1: Summary of number of graphs of $ext(C_4, n)$ (Part I)

edge number e	9	10	11	12	13	14	15	16	17	total
2	0	0	0	0	0	0	0	0	0	2
3	0	0	0	0	0	0	0	0	0	4
4	0	0	0	0	0	0	0	0	0	8
5	0	0	0	0	0	0	0	0	0	18
6	0	0	0	0	0	0	0	0	0	44
7	5	0	0	0	0	0	0	0	0	117
8	72	31	5	0	0	0	0	0	0	351
9	278	289	197	74	10	0	0	0	0	1230
10	529	883	1156	1097	670	222	34	2	0	5069
11	0	0	0	0	0	0	0	0	0	0

Table 5.2: Summary of number of graphs of $ext(C_4, n)$ (Part II)

Since the Ramsey number $r_3(C_6) = 12$, which means that if the complete graph has twelve or more vertices, then there must be a monochromatic cycle C_6 no matter how three colors we use to color the edges, hence, knowledge of the distribution of graphs on 11 vertices not containing C_6 is important.

n	$ext(C_4, n)$	number of graphs
2	1	1
3	3	1
4	4	1
5	6	1
6	7	4
7	9	5
8	11	5
9	13	10
10	16	2
11	23	3

Table 5.3: Summary of extremal graph numbers $ext(C_4, n)$

By executing the same programs $ext(C_6, 11) = 23$ is verified. The detailed results of $ext(C_6, 11)$ are given below, which is consistent with Yang's paper [18].

```
    1 graphs with 11 vertices and   0 edges
    1 graphs with 11 vertices and   1 edges
    2 graphs with 11 vertices and   2 edges
    5 graphs with 11 vertices and   3 edges
   11 graphs with 11 vertices and   4 edges
   26 graphs with 11 vertices and   5 edges
   66 graphs with 11 vertices and   6 edges
  168 graphs with 11 vertices and   7 edges
  442 graphs with 11 vertices and   8 edges
 1172 graphs with 11 vertices and   9 edges
 2998 graphs with 11 vertices and  10 edges
 7136 graphs with 11 vertices and  11 edges
14825 graphs with 11 vertices and  12 edges
24622 graphs with 11 vertices and  13 edges
30422 graphs with 11 vertices and  14 edges
27525 graphs with 11 vertices and  15 edges
18976 graphs with 11 vertices and  16 edges
```

```
 10487 graphs with 11 vertices and  17 edges
  4814 graphs with 11 vertices and  18 edges
  1861 graphs with 11 vertices and  19 edges
   596 graphs with 11 vertices and  20 edges
   147 graphs with 11 vertices and  21 edges
    26 graphs with 11 vertices and  22 edges
     3 graphs with 11 vertices and  23 edges
146332 graphs altogether
```

5.2 4-Good 3-colorings by Algorithms *DE* and *MR*

Since the *DE* algorithm is an iterative point by point algorithm which generates C_k-free 3-colorings of K_n $(n = 3, 4, \ldots)$ from the original two vertices by performing exhaustive enumerations, hence, to generate all 4-good 3-colorings of K_n $(n \leq 10)$, it is necessary to know the initial case of 4-good 3-coloring of K_2. All three 4-good 3-colorings on 2 vertices in mc-format were created by specialized program *gensample*.

By executing *DE* algorithm, the summary of all 4-good 3-colorings of K_n $(2 \leq n \leq 11)$ is given in Table 5.4.

number of vertices n	number of all 4-good 3-colorings
2	3
3	3
4	11
5	68
6	715
7	7580
8	38761
9	34009
10	1000
11	0

Table 5.4: Summary of number of good 3-colorings on K_n

An example of an adjacency matrix of a good 3-coloring of K_{10} is given below. The color degree sequences for each vertex are given in hexadecimal

below the matrix. For example, vertex 2 has two edges of color 1, three edges of color 2 and four edges of color 4.

```
Coloring on 10 vertices with 3 colors
 231232231
2 23311233
32 3123122
133 323211
2313 32121
31223 1321
213321 312
2212133 13
33212211 3
132111233
color 1 degs: 2223333334,  14 edges,   3 triangles
color 2 degs: 4342333332,  15 edges,   3 triangles
color 3 degs: 3434333333,  16 edges,   3 triangles
```

In Table 5.4, we can see that there is no 4-good 3-coloring for K_{11}, which verifies the upper bound for $r_3(C_4)$ is 11.By using utility program *shortmc*, we can obtain that there are only three possibilities for the color distribution on the edges which is shown in Table 5.5. The degrees of each color of K_{10} are less than or equal to 4 and the maximal number of edges is 16, which is consistent with the result from Bialostocki [1] and Clapham [4].

three possibilities	color 1	color 2	color 3
case 1	15	15	15
case 2	15	16	14
case 3	16	16	13

Table 5.5: Three good color distribution of 3-coloring on K_{10}

In this project, another method was employed to verify all 3-colorings of K_{10} obtained from *DE* algorithm are good 3-colorings. 3000 graphs with

number of edges 13 through 16 were obtained by choosing one color from all 3-colorings of K_{10} each time. On the other hand, four sets of graphs which are all on 10 vertices and no C_4 with number of edges 13, 14, 15, and 16 respectively from section 5.1 are given in Table 5.6.

number of edges	number of graphs
13	670
14	222
15	34
16	2
	total number: 928

Table 5.6: Number of C_4-free graphs on 10 vertices

The following shows that the graph counts of Table 5.6 by degree sequence. The three columns represent the number of edges, degree sequence, and the number of graphs.

```
Count graphs by degree sequence:


13 1122223346 (1)
13 1122223355 (1)
13 1122223445 (3)
13 1122224444 (1)
13 1122233336 (3)
13 1122233345 (20)
13 1122233444 (19)
13 1122333335 (8)
13 1122333344 (36)
13 1123333334 (10)
13 1133333333 (1)
13 1222222238 (1)
13 1222222247 (1)
13 1222222256 (1)
13 1222222229 (1)
```

```
 13 1222222337 (1)
 13 1222222346 (5)
 13 1222222355 (1)
 13 1222222445 (6)
 13 1222223336 (9)
 13 1222223345 (36)
 13 1222223444 (20)
 13 1222233335 (40)
 13 1222233344 (101)
 13 1222333334 (91)
 13 1223333333 (19)
 13 2222222255 (2)
 13 2222222237 (2)
 13 2222222336 (7)
 13 2222222345 (15)
 13 2222222444 (2)
 13 2222223335 (21)
 13 2222223344 (53)
 13 2222233334 (85)
 13 2222333333 (37)
 13 0222233444 (2)
 13 0222333335 (1)
 13 0222333344 (4)
 13 0223333334 (2)
 13 0233333333 (1)
Total: 40 sequences for 670 graphs

 14 1222233445 (5)
 14 1222234444 (2)
 14 1222333336 (1)
 14 1222333345 (8)
 14 1222333444 (15)
```

```
 14 1223333335 (4)
 14 1223333344 (18)
 14 1233333334 (6)
 14 1333333333 (1)
 14 2222223346 (1)
 14 2222223445 (2)
 14 2222224444 (2)
 14 2222233336 (4)
 14 2222233345 (21)
 14 2222233444 (16)
 14 2222333335 (10)
 14 2222333344 (58)
 14 2223333334 (37)
 14 2233333333 (11)
Total: 19 sequences for 222 graphs

 15 2222244444 (1)
 15 2222333445 (1)
 15 2222334444 (3)
 15 2223333336 (1)
 15 2223333345 (3)
 15 2223333444 (7)
 15 2233333344 (11)
 15 2333333334 (4)
 15 3333333333 (3)
Total: 9 sequences for 34 graphs

 16 3333333344 (2)
Total: 1 sequences for 2 graphs
```

By running the utility program *shortmc* with option *-v*, we obtained Table 5.7 which shows for each color of all 4-good 3-colorings of K_{10}, there

is a graph set S containing 1000 graphs which can be divided into num sub-sets. Each entry in Table 5.7 is identical with the results by running utility program *labely* with *sort -u* for graph set S. All graphs in each sub-set are isomorphic to each other and corresponding to one graph in Table 5.6 only. Suppose a set T is consist of all the graphs in Table 5.6, we conclude that there exists a many-to-one mapping between two graph sets S and T. Therefore, we verify that all 3-colorings of K_{10} generated by *DE* algorithm are 4-good 3-colorings. For each color, there exists only one isomorphism class which contains a large number of graphs isomorphic to each other. The number of graphs are 131, 125 and 124, respectively. However, for the other isomorphism classes, the number of isomorphic graphs reduces significantly. Hence, we can see that in graph set T, only one graph appears with very high frequency in graph set S, some graphs with lower frequency, and most of them never appear in S. The three graphs which appear frequently for color 1, 2 and 3 are isomorphic to each other and being one of the extremal graph of $ext(C_4, 10)$, which is given in Figure 5.2.

	number of isomorphism classes num
color 1	105
color 1	117
color 1	121

Table 5.7: Isomorphism analysis of graphs generated by choosing one color from all 3-colorings of K_{10}

In this project, another algorithm, *MR*, was utilized to generate all 4-good 3-colorings of $r_3(C_4)$. Due to $ext(C_4, 10) = 16$ and since the size of K_{10} is 45, there are only three possibilities of color distribution on edges given in Table 5.5. The number of graphs of case$i$-color$j$ for $1 \leq i \leq 3$ and $1 \leq j \leq 3$ is given in Table 5.8. Numbers in parenthesis represent the number of edges for each case$i$-color$j$. According to the basic idea of *MR* in Section 3.3.2, to generate all 4-good 3-colorings of K_{10}, for each case in Table 5.8, one graph of color 1 and one graph of color 2 are overlapped if there is no edge conflict between them; then the graphs of color 1 and color 2

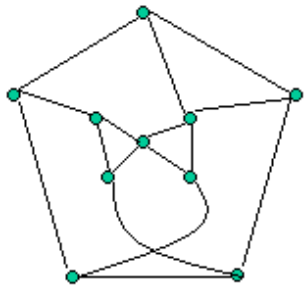

Figure 5.2: Graph with high appearing frequency in Table 5.7

are put in K_{10} and the existence of cycle C_4 is checked in the remaining part of K_{10} assigned color 3. The 3-coloring is good if no C_4 exists in color 3. To avoid skipping any good 3-coloring of K_{10}, for the color 2, all graphs without C_4 on 10 vertices with specified number of edges including isomorphism are applied. For color 1, we don't have to consider isomorphism between graphs since the isomorphic graphs of color 1 overlapped with one graph of color 2 will generate duplicate colorings. Therefore, all C_4-free graphs with edge number 13 through 16 and all isomorphic C_4-free graphs with edge number 15 and 16 are necessary and given in Table 5.9.

	color 1	color 2	color 3
case 1	34 (15)	34 (15)	34 (15)
case 2	34 (15)	2 (16)	222 (14)
case 3	2 (16)	2 (16)	670 (13)

Table 5.8: Three good color distributions with the corresponding number of graphs of K_{10}

To reduce the time and space complexity in generating all 4-good 3-colorings of K_{10} by *MR*, for each case in Table 5.8, all graphs in color 1 are read in and stored in memory. Then each time the program reads in one graph from the isomorphic set of color 2 and check if there is an edge confliction between it and all graphs in color 1 until all the graphs are

number of edges e	number of graphs	number of all isomorphic graphs
16	2	1360800
15	34	72817920
14	222	not generated
13	760	not generated

Table 5.9: Summary of number of graphs of $ext(C_4, 10)$

checked. Instead of storing all graphs with isomorphism in memory, the program only stores the colorings without edge conflict which reduces the space complexity significantly. After all the graphs in isomorphic set are eventually searched, the existence of cycle C_4 is checked for color 3.

By employing this algorithm, we obtain 1000 4-good 3-colorings of K_{10} which are identical with results from algorithm DE. Algorithms DE and MR are totally independent paths to obtain the same results, which is a very strog proof of the number of 4-good 3-colorings of K_{10} be 1000. For all the 4-good 3-colorings of K_{10}, none extends to C_k-free coloring of K_{11}, so $r_3(C_4) = 11$.

5.3 Lower Bounds of $r_3(C_k)$

The good lower bounds on Ramsey numbers $r_3(C_k)$ for $k = 5$, 6, and 7 are verified by the *TM* algorithm in this project. At the same time, lower bounds for Ramsey numbers with larger cycles, C_8 and C_{10}, are obtained first time as well. Since the space complexity of *TM* algorithm is small, it is possible to study the Ramsey numbers with larger cycles by this algorithm. However, the heuristic *TM* algorithm can only obtain the lower bounds for Ramsey numbers instead of accurate values. With more trajectories attempted, more accurate lower bounds on Ramsey number are obtained. The results of *TM* algorithm is given in Table 5.10.

Ramsey number	$r_3(C_5)$	$r_3(C_6)$	$r_3(C_7)$	$r_3(C_8)$	$r_3(C_{10})$
Lower bound	17	12	25	15	18

Table 5.10: Summary of lower bounds of $r_3(C_k)$

Chapter 6

Conclusions and Future Work

In this project, we conclude that there exists 1000 4-good 3-colorings for complete graph K_{10} by two different algorithms: *DE* and *MR*. Extremal graph numbers $ext(C_4, 10) = 16$ and $ext(C_6, 11) = 23$ were verified by computer programs. $r_3(C_5) \geq 17$, $r_3(C_6) \geq 12$, and $r_3(C_7) \geq 25$ were verified by the trajectory algorithm. We also conclude that the lower bound for $r_3(C_8)$ and $r_3(C_{10})$ are 15 and 18 respectively.

Although the *DE* and *MR* algorithms can verify the Ramsey number values theoretically, due to the large time and space complexity, it's not feasible to verify the values of Ramsey numbers with larger cycle by these two algorithms, therefore, we need to find more efficient algorithms to deal with the Ramsey numbers, such as $r_3(C_5)$, $r_3(C_6)$, and $r_3(C_7)$.

Bibliography

[1] A. Bialostocki and J. Schönheim, *On Some Turán and Ramsey Numbers for* C_4, Graph Theory and Combinatorics (ed. B. Bollobás), Academic Press, London, (1984) 29-33.

[2] B. Bollobás, *Extremal Graph Theory*, Academic Press Lomdon, 1978.

[3] B. Bollobás, *Graph Theory*, Springer-Verlag, 1979.

[4] C. R. J. Clapham, *The Ramsey Number* $r(C_4, C_4, C_4)$, Periodica Mathematica Hungarica, Vol. 18 (4), 1987, 317-318.

[5] C. R. J. Clapham, A. Flockhart and J. Sheehan, *Graphs Without Four-Cycles*, Journal of Graph theory, Vol. 13, No. 1, 1989, 29-47.

[6] R. Diestel, *Graph Theory*, Springer-Verlag, 1996.

[7] R. Faudree, A. Schelten and I. Schiermeyer, *The Ramsey Number* $r(C_7, C_7, C_7)$, Discussiones Mathematica Graph Theory, Vol 23, 2003.

[8] S. E. Fettes, R. L. Kramer, and S. P. Radziszowski *An Upper Bound of* 62 *On the Classical Ramsey Number* $R(3, 3, 3, 3)$, to appear in Ars Combinatoria, (2004).

[9] S. E. Fettes, *On the Classical Ramsey Number* $R(3, 3, 3, 3)$, M.S.Thesis Computer Science Department, RIT, 2001.

[10] R. E. Greenwood and A. M. Gleason, *combinatorial Relations and chromatic Graphs*, Canadian Journal of Mathematics, 7 (1955), 1-7.

[11] T. Łuczak, $R(C_n, C_n, C_n) \leq (4+o(1))n$, Journal of Combinatorial Theory, Series B, 75 (1999), 174-187.

[12] B. McKay, *nauty: a set of procedures for determining the automorphism group of a graph and optionally for canonically labeling it*, http://cs.anu.edu.au/~bdm/nauty.

[13] K. Piwakowski, S. P. Radziszowski, *Towards the Exact Value of the Ramsey Number* $R(3, 3, 4)$, Congressus Numerantium, 148 (2001), 161-167.

[14] S. P. Radziszowski, K. Tse, *A Computational Approach for the Ramsey Numbers* $R(C_4, K_n)$, Journal of Combinatorial Mathematics and Combinatorial Computing, 42 (2002), 195-207.

[15] S. P. Radziszowski, *Small Ramsey Numbers*, Electronic Journal of Combinatorics, 1 (1994), revision #9, 2002, http://www.combinatorics.org/Surveys.

[16] Y. Yang, P. Rowlinson, *On the third Ramsey numbers of graphs with five edges*, Journal of Combinatorial Mathematics and Combinatorial Computing, 11 (1992), 213-222.

[17] Y. Yang, P. Rowlinson, *On extremal graphs without* 4*-cycles*, Utilitas Mathematica, 41 (1992), 204-210.

[18] Y. Yang, P. Rowlinson, *On graphs without 6-cycles and related Ramsey Numbers*, Utilitas Mathematica, 44 (1993), 192-196.

[19] Y. Yang, P. Rowlinson, *The third Ramsey numbers for graphs with at most four edges*, Discrete Mathematics, 125 (1994), 399-406.

[20] Y. Yang, P. Rowlinson, *On the third Ramsey numbers of graphs with six edges*, Journal of Combinatorial Mathematics and Combinatorial Computing, 17 (1995), 199-208.