Master's Project Proposal: The Study of Ramsey Numbers $r(C_k, C_k, C_k)$

Yan Li
Department of Computer Science
Rochester Institute of Technology

September 5, 2003

Abstract

The Ramsey number $r(C_k, C_k, C_k)$ 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, I will overview all literature on the Ramsey numbers $r(C_k, C_k, C_k)$, and I will attempt to improve our knowledge on this subject. In particular, first, algorithms to check if a graph G contains any specific path or cycle and to construct extremal graphs for a cycle with k vertices will be developed. Second, multicolored graphs will be constructed to verify the Ramsey number values $r_3(C_3)$ and $r_3(C_4)$. The lower bounds for the Ramsey numbers $r_3(C_5)$, $r_3(C_6)$, and $r_3(C_7)$ will be provided as well. Additionally, I will search for the possibility of further research for larger k, especially for $r_3(C_8)$ and $r_3(C_{10})$. Most of the results will be based on computer algorithms.

Contents

1 Introduction

A *graph*, $G = (V(G), E(G))$, is a topological 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. In this project, we only consider undirected finite graph without any loops and multiple edges. A *complete graph* K_n is a graph on n vertices with edges connecting any pair of vertices. A *path* P_k defined on k vertices $V(P_k) = \{x_1, x_2, \ldots, x_k\}$ is the set of edges $E(P_k) = \{x_1x_2, x_2x_3, \ldots, x_{k-1}x_k\}$. The length of a path is the number of edges traversed. A *cycle* C_k defined on k vertices $V(C_k) = \{x_1, x_2, \ldots, x_k\}$ is the set of edges $E(C_k) = \{x_1x_2, x_2x_3, \ldots, x_{k-1}x_k, x_kx_1\}$. An *edge coloring* $(r_1, r_2, \ldots, r_m)$, $r_i \geq 1$ for $1 \leq i \leq m$, is an assignment of one of m colors to each edge of graph G, such that there are r_i edges in color i.

The *classical Ramsey number* $r(r_1, r_2, \ldots, r_m)$ is defined to be the least integer $n > 0$ such that for a complete graph K_n with any edge coloring method $(r_1, r_2, \ldots, r_m)$, there always exists at least one monochromatic complete subgraph K_{r_i} in color i for $1 \leq i \leq m$. However, the classical Ramsey numbers are too complicated to be studied by this project. So in this project we concentrate on a simpler 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.

A good and detailed overview of known bounds and exact values of various types of Ramsey numbers is given in Radziszowski's survey "Small Ramsey Numbers"[13]. The classical Ramsey numbers with 2-coloring have been studied thoroughly and many results have been obtained. However, in the multicolor case $m > 2$, 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 $r(3, 3, 3) = 17$. Up to now for the special case of Ramsey numbers of the form $r_3(C_k)$, there are only six known exact values shown in Table 1. Obviously, $r_3(C_3)$ and $r_3(K_3)$ are the same case.

2 Proposal

We will overview all literature on the Ramsey numbers $r_3(C_k)$, and attempt to improve our knowledge on it by designing, implementing and executing computer algorithms. This will be approached step by step. First, we will

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

Table 1: Values of Ramsey numbers of form $r_3(C_k)$

develop the *path search method* and *cycle search method* to check if a graph G contains any specific monochromatic path or cycle. Based on this algorithm, we can develop the *C_k-free graph construction method* to construct C_k-free graphs and find the extremal graph numbers. Second, $r_3(C_3) = 17$ and $r_3(C_4) = 11$ will be verified with *direct extension algorithm*, when multicolored graphs in mc-format are constructed. By using C_k-free graph construction method and the obtained extremal graph numbers, *matching algorithm* will be employed to verify $r_3(C_4) = 11$. Finally, the *trajectory algorithm* with random walk will be applied to verify the lower bounds of $r_3(C_5) \geq 17$, $r_3(C_6) \geq 12$, and $r_3(C_7) \geq 25$. By this algorithm, we will also find the good lower bounds of Ramsey number values of $r_3(C_8) \geq 15$ and $r_3(C_{10}) \geq 18$ at the first time.

2.1 Approach

2.1.1 Isomorphism and Extremal Graph Number

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 notion 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, where one-to-one mapping means, $\{u, v\} \in E(G)$ 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 doesn't contain a subgraph isomorphic to H [6]. In the project we only consider H as a cycle C_k.

2.1.2 Path search method (PS)

The PS method checks if a graph G contains any specific monochromatic 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 with k vertices found in graph G; 0 if no path with 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; G, graph.
  output: boolean. 1 if a path with k vertices found between vertices v_1 and v_2; 0 if no path with 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: find 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: find 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: find 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.

2.1.3 Cycle search method (CS)

The CS method checks if there exists a 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 with k vertices found in graph G; 0 if no cycle with 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
```

2.1.4 C_k-free graph construction method (FG and FGNOISO)

The FG and $FGNOISO$ methods are applied to create all isomorphic and nonisomorphic C_k-free graphs on n vertices by generating all graphs on n vertices and eliminating the graphs with cycle C_k based on the results of cycle searching. The nonisomorphic graphs can be obtained by checking the canonical labelings of all graphs and removing the identical labelings by option sort -u. Please see 2.3 for canonical labeling.

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
```

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

procedure:

```
FGNOISO(n)
graph_set_1 <- FG(n)
graph_set_2 <- LABELY(graph_set_1)
graph_set_3 <- SORTU(graph_set_2)
return graph_set_3
```

- 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.

- LABELY(*graph_set*)
  input: *graph_set*, a graph set containing all graphs on n vertices.
  output: a graph set containing all graphs on n vertices and with the

same canonical labeling for the isomorphic ones.
procedure: label the graph_set to get the identical labelings for the isomorphic ones.

- SORTU(*graph_set*)
  input: *graph_set*, a graph set containing all graphs on n vertices and with the same labelings for the isomorphic ones.
  output: a graph set containing only nonisomorphic graphs on n vertices.
  procedure: remove the graphs with identical labelings from graph_set.

The following two tables show results of the number of nonisomorphic and isomorphic C_k-free graphs.

number of vertices	8	9	10
no C_3	410	1897	12172
no C_4	351	1230	5069
no C_5	929	3727	17565
no C_6	1541	6641	30247
no C_7	3139	15238	79159

Table 2: The number of nonisomorphic C_k-free graphs

number of vertices	8	9	10
no C_3	1762	9552	62340
no C_4	1616	6318	28640
no C_5	3657	17133	85762
no C_6	6058	28371	141213
no C_7	11176	60538	342829

Table 3: The number of C_k-free graphs

2.2 Algorithms for (C_k, C_k, C_k)-coloring

Based on the definition of Ramsey number, the most direct way of computing the Ramsey number $r_3(C_k)$ is generating all the 3-colorings of K_n with $n = 1, 2 \ldots$, removing the graphs with cycle C_k, and finding the first with empty output. However, due to the large time and space complexity, this naive method is not feasible for even moderate n. In the following, we will outline three algorithms to construct the C_k-free 3-colorings of the edges of K_n.

2.2.1 Direct extension algorithm (DE)

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 originally two vertices by performing exhaustive enumerations. By this method, $r_3(C_3) = 17$ and $r_3(C_4) = 11$ will be verified.

DE(n, k)
input: n, number of vertices of K_n; k, number of vertices of cycle.
output: a set of C_k-free 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 all C_k-free 3-colorings of edges of complete graph on 3 vertices.

- ADDV(*graph*)
  input: a 2-coloring of K_n.
  output: a set of all 3-colorings of K_{n+1}.
  procedure: generate all 3-colorings of edges of complete graph on $n+1$ vertices by performing exhaustive enumerations.

- ADDG(*coloring_set*, *coloring*)
  input: *graph_set*, set of C_k-free 3-colorings of edges of K_n; *graph*, 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-colorings of K_n to the previous set.

However, the exponentially increasing number of graphs makes it almost impossible to verify the Ramsey numbers with cycle larger than C_6 even if the temporal complexity is linearly dependent on the number of vertices n.

2.2.2 Matching algorithm (MR)

The matching algorithm is used to generate (C_k, C_k, C_k) colorings from the C_k-free graphs based on the results of nonisomorphic and isomorphic C_k-free graphs. $r_3(C_4) = 11$ will be verified by this method. In order to generate all C_k-free 3-colorings of K_{10}, if none extends to C_k-free coloring of K_{11}, then $r_3(C_4) = 11$. This verifies the result by Bialostocki [1] and Clapham [3]. Due to $ext(4, 10) = 16$ and the number of edges of complete graph on 10 vertices is 45, we only consider three cases of the color distribution on edges, which is shown in the following table.

MR(n, k)
input: n, number of vertices of graph; k, number of vertices of cycle.
output: set of C_k-free 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] <- FGNOISO(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
```

three cases	color1	color2	color3
case 1	15	15	15
case 2	14	15	16
case 3	13	16	16

Table 4: The three color distribution cases of $r_3(C_4)$

- EXT(k, n)
  input: k, number of vertices of cycle; n, number of vertices of final 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 final graph.
  output: an 3×m array denoting all the possible distributions of three colors of a graph.

procedure: generate a 2_D array to store the color distribution.

- CONFLICT(g_1, g_2)
  input: g_1, g_2, two monochromatic C_k-free graphs.
  output: boolean. 1 if there is an edge confliction between two graphs; 0 if no edge confliction exists.
  procedure: check edge confliction between two graphs.

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

- MULTI(g_1, g_2, g_3)
  input: three monochromatic C_k-free graphs.
  output: a C_k-free 3-colorings of K_n.
  procedure: combine the three monochromatic graphs to obtain a C_k-free 3-colorings of K_n.

2.2.3 Trajectory algorithm (TM)

The TM algorithm is a random version of DE algorithm which is heuristic, not exhausting, but can establish lower bounds for Ramsey numbers. The idea is choosing a cycle-free 3-colorings of K_n randomly, generating more cycle-free 3-colorings of K_{n+1} recursively until no cycle-free graphs, and repeating the trajectory generating procedure as much as possible. A computing chain of multicolored graphs with vertices $2, 3, 4, \ldots$ is regarded as a trajectory and more trajectories attempted, closer to the result of DE algorithm can be obtained. By this method, $r_3(C_5) \geq 17$, $r_3(C_6) \geq 12$, $r_3(C_7) \geq 25$, $r_3(C_8) \geq 15$, and $r_3(C_{10}) \geq 18$ will be found.

TM(k, n)
input: k, integer > 1, number of vertices of cycle; n, number of vertices of graph.

output: a C_k-free 3-colorings 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 no hard disk storage space and has the same computational complexity as DE. However, at any time, it can only obtain the lower bound, instead of the exact values of Ramsey numbers. When more and more trajectories are attempted, with an increasing certainty, a number will exactly be the value of Ramsey number. To increase the efficiency of this method, enhanced algorithm is implemented to make the trajectories go farther.

2.3 Software tools

There are two formats used to represent the adjacency matrix of a given graph. Y-format is an $ASCII$ format and the lower 6 bits of a byte are used to maintain the given adjacencies. Mc-format is used to store and manipulate multicolored graphs, which allows each graph coloring to be represented by one line with two security bits in each byte. For two graphs G_1 and G_2, they are isomorphic if and only if canlab(G_1) = canlab(G_2). Here, the function *canlab* is produced by *labely* for the graphs in y-format. In this project, program $LABELY$, developed by B. McKay and computing a canonical labeling of graphs, was applied [10].

C++ is chosen as the software developing language for its apparent advantages of object-oriented features. Eight classes are designed by the author.

- $YFormat$ – handling the operations on a y-format string.

- *MCFormat* – handling the operations on a mc-format string that represents a graph with multicolor edges.
- *Combine* – generating m combinatorial numbers from n given numbers, namely all of the combinatorial values of $C(n, m)$.
- *Path* – recording the path between each pair of vertices.
- *Ramsey* – generating 3-color, n vertices and cycle-free completed graphs according to cycle free monochromatic graphs with the same vertex number.
- *MCGraph* – handling the operations on a multicolor graph in mc-format.
- *Graph* – installing the undirected non-weight graph matrix with y-format.
- *Trajectory* – finding Ramsey number values by throwing lots of trajectories and determine the farthest one, and the smallest graphs are with 2 vertices.

3 Deliverables

- project report
  All literature on the Ramsey numbers $r_3(C_k)$ were overviewed. The Ramsey number values $r_3(C_3)$ and $r_3(C_4)$ and the lower bounds for $r_3(C_5)$, $r_3(C_6)$, and $r_3(C_7)$ were verified. Additionally, the lower bounds for $r_3(C_8)$ and $r_3(C_{10})$ were obtained.
- experiment
  Developing the path search and cycle search methods to check if a graph contains any specific monochromatic path or cycle. Developing the C_k-free graph construction method to construct C_k-free graphs and find the extremal graph numbers. Implementing three algorithms to verify the values of $r_3(C_k)$ and obtain the lower bounds for them.
- source code
- result
  $r_3(C_3) = 17$ and $r_3(C_4) = 11$ were verified with direct extension algorithm. $r_3(C_4) = 11$ was also verified by matching algorithm with

C_k-free graph construction method and the obtained extremal graph numbers. $r_3(C_5) \geq 17$,$r_3(C_6) \geq 12$, and $r_3(C_7) \geq 25$ were verified by the trajectory algorithm. Also, $r_3(C_8) \geq 15$ and $r_3(C_{10}) \geq 18$ were obtained at the first time.

4 Timetable

The literature search and background reading began in Mar. 2001 and was completed by May. 2001. The software development was completed by the end of Sep. 2001 at RIT. Writing up the project and defending should take place by the end of December. 2003.

Item	Date
Literature Search	March 2001
Preliminary Reading	April 2001
Software Developement	September 2001
Proposal Filed	July 2003
Project Write-up	September 2003
Project Defense	December 2003

Table 5: Timetable of the project

References

[1] A. Bialostocki and J. Schönheim, *On Some Turán and Ramsey Numbers for* C_4, Graph Theory and Combinatorics.

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

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

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

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

[6] R. Faudree, A. Schelten and I. Schiermeyer, *The Ramsey Number* $r(C_7, C_7, C_7)$, appeared.

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

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

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

[10] 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.

[11] K. Piwakowski, S. P. Radziszowski, *Towards the Exact Value of the Ramsey Number* $R(3, 3, 4)$, appeared.

[12] S. P. Radziszowski, K. Tse, *A Computational Approach for the Ramsey Numbers* $R(C_4, K_n)$, to appear.

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

[14] 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.

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

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

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

[18] 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.