AN ANT COLONY OPTIMIZATION TECHNIQUE FOR SOLVING GRAPH COLORING PROBLEM

(A MAJOR PROJECT REPORT)

Submitted By

Jayashree Mahapatra, Univ. Roll No. - 21301217074, Univ. Reg. No. – 172131010047

Arijit Ray, Univ. Roll No. -21301217105, Univ. Reg. No. – 172131010016

Sankha Misra, Univ. Roll No. - 21301217041, Univ. Reg. No. – 172131010080

Bhagyashree Maity, Univ. Roll No. - 21301217098, Univ. Reg. No. - 172131010023

Under the Supervision of

Prof. Arnab Kole

Assistant Professor

In partial fulfillment for the award of the degree

Of

BACHELOR OF COMPUTER APPLICATION

THE HERITAGE ACADEMY

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

2020

ACKNOWLEDGEMENT

We would take the opportunity to thank ***Prof. (Dr.) Gour Banerjee, Principal, The Heritage Academy*** for allowing us to form a group of 4 people and for supporting us with the necessary facilities to make our project worth.

We are thankful to ***Prof. Arnab Kole (Assistant Professor, BCA)*** our ***Project Guide*** who constantly supported us, and ***Prof. Avik Mitra (Assistant Professor, BCA), the Project Coordinator*** for providing and clarifying the administrative formalities related to project proceedings. Their words and instructions have assisted us to excel in our project.

We thank all our other faculty members and technical assistants at The Heritage Academy for paying a significant role during the development of the project. Last but not the least, we thank all our friends for their cooperation and encouragement.

Signature: ___________________________________

(Arijit Ray)

Signature: ___________________________________

(Bhagyashree Maity)

Signature: ___________________________________

(Jayashree Mahapatra)

Signature: ___________________________________

(Sankha Misra)

The Heritage Academy

Kolkata

PROJECT CERTIFICATE

This is to certify that the following students:

Name of the students	Roll No.	Registration No.
1.JAYASHREE MAHAPATRA	21301217074	172131010047
2.ARIJIT RAY	21301217105	172131010016
3.SANKHA MISRA	21301217041	172131010080
4.BHAGYASHREE MAITY	21301217098	172131010023

of 3rd Year 2nd Semester in BCA(H) have successfully completed their Major Project Work on

An Ant Colony Optimization Technique for Solving Graph Coloring Problem

towards *partial fulfillment* of Bachelor of Computer Applications from Maulana Abul Kalam Azad University of Technology, West Bengal in the year **2019-2020**.

Prof.(Dr.)Gour Banerjee Principal The Heritage Academy	Prof. Arnab Kole Project Guide Assistant Professor Department of BCA The Heritage Academy	Prof. Avik Mitra Project Coordinator Assistant Professor Department of BCA The Heritage Academy

ABSTRACT

Graph coloring plays an important role in the field of graph theory. Though various advancements have been made in the arena of modern computer algorithms, still graph coloring algorithms are well known research topic in Mathematics and Computer Science around the world. Meta-heuristic Search Algorithms are very efficient optimization techniques when it comes to problems that require large solution space. Instead of giving exact solution, it tries to produce an approximate one that takes less time. These algorithms are problem independent and hence can be applied to a wider variety of problems. Some popular meta-heuristic algorithms in Evolutionary Algorithms are Genetic Algorithms, Simulated Annealing, Ant Colony Optimization etc. One of the most notable applications of these algorithms is graph coloring problem as this kind of problem often involves a huge search space due to which the exact solution of this problem is difficult to obtain. Since the working principle of evolutionary algorithms is based on the use of different heuristics in probabilistic manner, it can produce optimal solution for graph coloring. In this project work, we have tried to solve Graph Coloring Problem using one of the metaheuristic techniques - Ant Colony Optimization.

TABLE OF CONTENTS

1. INTRODUCTION

It has already been shown that the decision version of graph coloring problem is NP complete **[1]** and the approximate version is NP hard **[2]**. Hence till date no deterministic algorithm has been proposed which can produce the optimal solution of this problem in polynomial time. It has also been shown over the years that heuristic approaches **[3, 4]** are worthy of use to produce optimal or near optimal solution for this type of problems. Graph coloring problem is meant to be assigning colors to the certain elements of a graph subject to certain constraints. Graph coloring is a special case of graph labeling. The most common graph coloring problem is Vertex coloring. The problem is, m colors are given, and we have to find a way of coloring the vertices of a graph in such a way that no two adjacent vertices are colored using the same color. A graph in which every vertex has been assigned a color according to a proper coloring is called a properly colored graph. A graph *G* that requires k different colors for its proper coloring, and no less, is called a *k*-chromatic graph and the number *k* is called the chromatic number of *G*. The following figures are the examples of proper, improper and chromatic coloring of a graph:

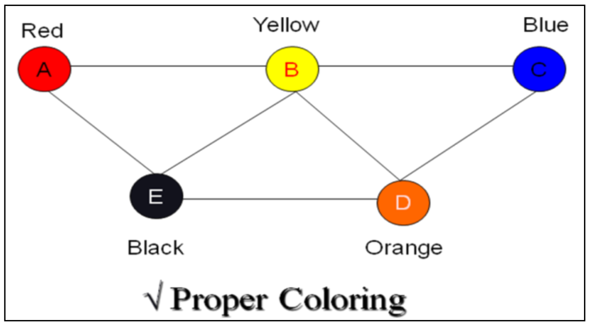

Figure 1: Proper Graph Coloring

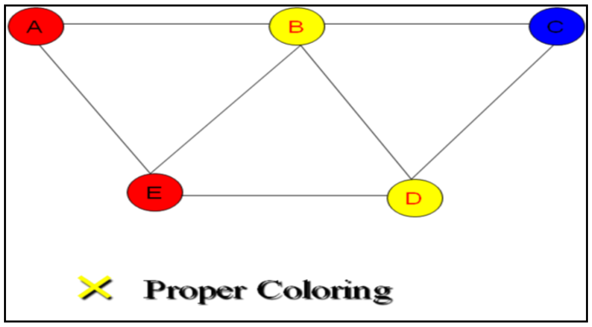

Figure 2: Improper Graph Coloring

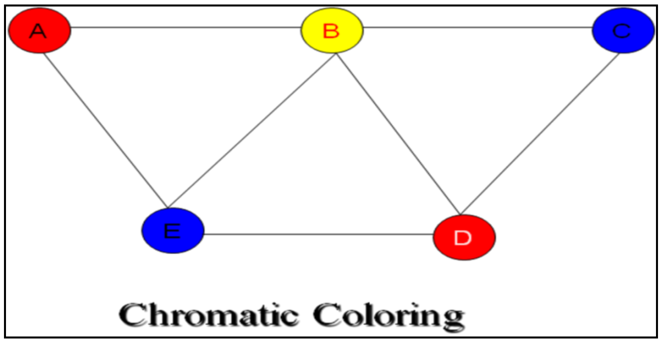

Figure 3: Chromatic Graph Coloring

2. EARLIER WORK

A variety of research works in the field of computer science and mathematics have been implemented to solve the theoretical areas of graph coloring. GCP has been solved using adjacency matrix and link list based exact algorithm by Shukla et. al. **[5, 6]**. Some direct approaches like Branch-and-Bound, Branch-and Cut and Backtracking algorithms **[7, 8, 9]** have been implemented to solve GCP. Graph coloring problem has been treated as Constraint Satisfaction Problem and one possible solution has been proposed by Diaz et. al. **[10]**. With the advent of Genetic algorithms and other evolutionary approaches, the solutions of Graph Coloring Problem and its applications have been achieved in optimal or nearly sub optimal time **[11, 12]**. An Ant Colony approach-based algorithm has been proposed in **[13, 14]** to solve GCP. Dahl **[15]** applied one Artificial Neural Networks approach to solve graph coloring problem by, and more recently the algorithm has been modified by Jagota **[16]**.

3. APPLICATIONS

There are many areas where graph coloring has been used to solve practical problems like timetable scheduling of schools and colleges **[17]**, examination scheduling of colleges and universities **[18]**, register allocation **[19]**, microcode optimization **[20]**, channel routing **[21],** printed circuit testing **[22],** allocation of electronic bandwidth **[23]**, the design and operation of flexible manufacturing systems **[24],** timetable scheduling **[25]** etc. where the evolutionary algorithms that use heuristic approaches have been proposed to produce optimal or suboptimal solution in a satisfactory amount of time.

4. ANT COLONY OPTIMIZATION

Ant colony optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems. Ant algorithms were first proposed by Dorigo et. al. **[26]** as a multi-agent approach to solve difficult combinatorial optimization problems such as Travelling Salesman Problem TSP **[27]** and Quadratic Assignment Problem (QAP). ACO algorithm has also been used to solve Constraint Satisfaction Problem **[28]** and the hybridization of ACO and Quantum Computing has been proposed for Link Prediction **[29]**. ACO algorithm is an iterative optimization technique. In each iteration, a number of ants (a set of software simulated agents) fabricate feasible solutions using heuristic information and they exploit the solution information of preceding batch of ants. These exploitations are individualized by a trail of digital pheromones, which is embedded on the individual segments of a solution. Small amounts are deposited during initialization while larger quantities are embedded, at the conclusion of each generation, in proportion to solution grade. Pheromone can be embedded on the constituents and/or the association used in a solution according to the problem. At each instance of newly generated solutions, each ant hand-picks probabilistically from the set of applicable constituents based on an amalgamation of the quantity of pheromone on that constituent and a heuristic value of that constituents' utility.

4.1. BASIC ACO PROCEDURE

- Initialize Pheromone Trail
- Repeat for each ant
    - Solution construction using pheromone trail.
    - Update pheromone trail.
- Until stopping criteria

5. PROPOSED ALGORITHM

ALGORITHM:

INPUT: Adjacency Matrix (n x n) where N is the number of vertices

STEPS:

```
STEP 1: Calculate maximum degree $\Delta$ (max_degree) of the input graph
STEP 2: Create one coloration using a single color (1) to all the vertices
STEP 3: Initialize q <- 1
STEP 4: While there exits at least one conflicts
        STEP 4.1: Distribute m ants to m vertices
        STEP 4.2: If $x^{th}$ ant discover a confliction locally
                STEP 4.2.1: If the color of $x^{th}$ vertex $\leq$ max_degree
                        Step 4.2.1.1: Change the color of $x^{th}$ vertex with color q + 1
                        Step 4.2.1.2: q <- q + 1
                STEP 4.2.2: Else
                        Step 4.2.2.1: Change the color of $x^{th}$ vertex with a random color
                        from the range of 1 to (max_degree + 1)
        STEP 4.3: Check global confliction
        STEP 4.4: If global confliction removed
                Step 4.4.1: Break cycle
STEP 5: Output a valid coloration
```

In the graph coloring problem, the pheromone concept of the real-world situation of ant colony has been implemented by the confliction of colors in different vertices. When a particular ant is visiting a vertex it will check for a conflict i.e. adjacent vertices have same colors, it's trying to remove the conflict by choosing another color from the set of available colors (max_degree[graph] + 1) if no conflict is found the color set doesn't change. From the result of those the local pheromone table is updated. For all the ants when they should update their local pheromone table; after coming outside the loop, the global pheromone table is then updated with the result of local pheromones.

6. RESULT AND DISCUSSION

The algorithm has been tested on some of the DIMACS **[30]** benchmarks graphs. The algorithm has been run in Microsoft's Visual Studio Code 1.45.1 and run on GCC-6.3.0-1 compiler with Operating System: Windows 10 Home 64 bit (10.0, Build: 18362), Processor: Intel Core i5-8250U CPU @ 1.6 GHz x 4 and RAM: 8 GB. The following table represents the detailed description of the result found by the algorithm.

Table 1: Obtained Chromatic Number & ACPU Time

Graph Instance	Vertex	Edge	Expected Chromatic Number	Chromatic Number Obtained	ACPU Time (Sec)
myciel3.col	11	20	4	4	0.001
myciel4.col	23	71	5	5	0.002
myciel5.col	47	236	6	6	0.02
games120.col	120	638	9	9	1.876
mugg88_1.col	88	146	4	4	0.508
le450_25d.col	450	17425	25	36	455.322
huck.col	74	602	11	11	0.165
jean.col	80	254	10	10	0.353
petersen.col	10	15	3	3	0.001
queen5_5.col	25	320	5	8	0.003
queen6_6.col	36	580	6	11	0.019
queen7_7.col	49	952	7	10	0.102
zeroin.i.3.col	206	3540	30	30	14.38
david.col	87	812	11	12	0.474
anna.col	138	986	11	12	1.021
3-Insertions_3.col	56	110	4	4	0.065
miles250.col	128	774	8	9	1.95
DSJC125.1.col	125	736	?	8	1.45
dsjc500.1.col	500	12458	?	20	541.85
dsjc500.9.col	500	224874	?	172	4398.12
school1.col	385	19095	?	42	282.578
school1_nsh.col	352	14612	?	39	190.119
1-FullIns_3.col	30	100	?	8	0.019
2-FullIns_3.col	52	201	?	10	0.123
1-FullIns_4.col	93	593	?	11	0.617

7. LIMITATION AND FUTURE SCOPE

7.1. LIMITATION

The ACOGCP produces optimal coloring in minimum possible time. Although, it fails to provide optimal chromatic number for each and every case, it tries to do a significant job of create a significant coloring with a minimum number of coloring. However, time of execution increases drastically in case dense graphs.

7.2. FUTURE SCOPE

To get further better results, we need to make modifications to the proposed algorithm. We might also use different metaheuristic approaches or different algorithms. We might also build a hybrid algorithm using genetic algorithm and simulated annealing. Other meta-heuristic algorithms like Tabu Search and Swarm Optimization techniques can also be tried.

8. CONCLUSION

The ACOGCP algorithm is indeed a simple yet efficient algorithm. The algorithm performs well when the graphs are not dense in nature. In fact, the best results are achieved when the cases where maximum degree (Δ) of the graph < (number of vertices/2). Beyond this criterion, the execution time increases exponentially, thus dropping its performance and making it somewhat problem specific. So, ACOGCP has difficulties reaching exact solutions sometimes, but it gives us a value close to the actual result. For some of the graphs where expected chromatic number has not been given in the benchmark sets, the algorithm finds the chromatic number but with a bit higher time.

9. REFERENCES

[1] Garey M.R. and Johnson D.S. Computers Intractability: A Guide to the Theory of NP-Cpmpleteness. W.H. Freedman and Co., 1979.

[2] Baase, S. and Gelder A.V., Computer Algorithms: Introduction To Design and Analysis. Addison-Wesley, 1999.

[3] Brelaz, D.: New methods to color vertices of a graph. Communications of ACM 22, 251-256 (1979).

[4] Hertz, A., De Werra, D.: Using tabu search techniques for graph coloring. Computing 39, 345-351 (1987).

[5] Ajay Narayan Shukla, M.L.Garg, An approach to solve graph coloring problem using adjacency matrix, Bioscience Biotechnology Research Communications, 12(2):472-477, June, 2019

[6] Ajay Narayan Shukla, Vishal Bharti, Madan L.Garg, A Linked List-Based Exact Algorithm for Graph Coloring Problem, International Information and Engineering Technology Association, Vol. 33, No. 3, pp. 189-195, June, 2019

[7] Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. INFORMS Journal on Computing, 8(4):344–354, 1996.

[8] Isabel Méndez-Diaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. Discrete Mathematics, 154(5):826–847, 2006.

[9] R. Monasson. On the analysis of backtrack procedures for the coloring of random graphs. In E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, Complex Networks, pages 235–254. Springer, 2004.

[10] Díaz, Isabel Méndez, and Zabala, Paula. 1999, A Generalization of the Graph Coloring Problem, Departamento de Computacion, Universidad de Buenes Aires.

[11] Ali, F. F., Nakao, Z., Tan, R. B., and Yen-Wei, Chen. 1999. An evolutionary approach for graph coloring. In Proceedings of The International Conference on Systems, Man, and Cybernetics, 527- 532. IEEE.

[12] Tagawa, K., Kanesige, K., Inoue, K., and Haneda, H. 1999. Distance based hybrid genetic algorithm: an application for the graph coloring problem. In Proceedings of Congress on Evolutionary Computation, 2332. Washington DC.

[13] Costa D., Hertz A. and Dubuis O.: “Ants Can Color Graphs”, Journal of the Operational Research Society, 48(1997), 295-305

[14] Anindya J. Pal, Biman Ray, Nordin Zakaria, Ken Naono & Samar Sarma, “Generating Chromatic Number of a Graph using Ant Colony Optimization and Comparing the Performance with Simulated Annealing”, Vol.50, pp 629-639, Procedia Engineering, 2012.

[15] E. D. Dahl, “Neural Networks algorithms for an NP complete problem: Map and graph coloring”, IEEE International Conference on Neural Networks, vol. 3, pp. 113-120.

[16] A. Jagota, “An adaptive, multiple restarts neural network algorithm for graph coloring”, European Journal of Operational Research, vol. 93, pp. 257-270, 1996.

[17] Wood D.C.: “A technique for coloring a graph applicable to large scale time-tabling problems”, Computer Journal, vol. 12, pp. 317-319, 1969.

[18] Leighton F.T., “A graph coloring algorithm for large scheduling problems”, Journal of Research of the National Bureau of Standards, vol. 84, no. 6, pp. 489-505, 1979.

[19] Chow F.C. and Hennessy J.L., “Register allocation by priority based coloring”, Proceedings of the ACM Sigplan 84 symposium on compiler construction New York, pp. 222-232, 1984.

[20] Micheli G. D.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, 1994.

[21] Sarma S.S, Mondal R. and A. Seth: “Some sequential graph coloring algorithms for restricted channel routing”, INT. J. Electronics, vol. 77, no. 1, pp. 81-93, 1985.

[22] A. Gamst, Some lower bounds for class of frequency assignment problems, IEEE Transactions on Vehicular Technology, vol. 35, no. 1, pp. 8-14, 1986.

[23] Micheli G.D., Synthesis And Optimization Of Digital Circuits. McGraw-Hill, 1994.

[24] S.S. Sarma, R. Mondal and A. Seth, “Some sequential graph coloring algorithms for restricted channel routing”, INT. J.Electronics, vol. 77, no. 1, pp. 81-93, 1985.

[25] Ramzi A. Haraty, Maram Assi, Bahia Halawi, “Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem”, Science Direct, Procedia Computer Science, Volume 126, Pages 899-906, 2018

[26] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of operating agents, IEEE Transactions on Systems, Man and Cybernatics-Part B, vol. 26, pp. 29-41, 1996

[27] M. Dorigo, L. M. Gambardella, Ant colony systems: A cooperative learning approach to the travelling salesman problem, IEEE Transactions on Evolutionary Computation, vol. 1, pp. 53-66, 1997

[28] Takuya Masukane, Kazunori Mizuno, Hiroto Shinohara, Ant Colony Optimization with Negative Feedback for Solving Constraint Satisfaction Problems, Conference on Technologies and Applications of Artificial Intelligence (TAAI), DOI**:** 10.1109 / TAAI.2018.00041, IEEE, December, 2018

[29] Zhiwei Cao, Yichao Zhang, Jihong Guan, Shuigeng Zhou, Link Prediction based on Quantum-Inspired Ant Colony Optimization, Scientific Reports volume 8, Article number: 13389, September, 2018

[30] https://mat.tepper.cmu.edu/COLOR/instances.html (accessed on April 18, 2020).

APPENDIX

```
#include<iostream>
#include<vector>
#include<unordered_set>
#include<ctime>
#include<fstream>
#include<stdlib.h>
#include<iomanip>
#include<numeric>
#include<stdexcept>
using namespace std;
class Graph //Defining a Graph class
{
   public:
      string filename;
      vector<vector <int>> edges;
      vector<vector <bool>> adjmat;
      int nvertex,nedges,max_degree;
      Graph(string str);
      void printAdjMat();
};
Graph :: Graph(string str)//Graph Constructor
{
   filename=str;
   ifstream inData(str);
   if(inData.fail())
   {
      cout<<"\nFilename doesn't exist!!....\n";
      exit(1);
   }
   int x,y;
   char e;
   string s;
   inData >> s;
   while(s != "edge")
         inData >> s;
   inData>>nvertex>>nedges;
   cout<<"\nFile name:"<<filename<<endl;
   cout<<"\nVertex:"<<nvertex<<endl;
   cout<<"Edges:"<<nedges<<endl;
   adjmat.resize(nvertex,vector<bool>(nvertex,0));
   edges.resize(nedges,vector<int>(2));
   for (int i=0; i<nedges; i++)
   {
        inData >> e >> x >> y;
        edges[i][0]=x;
        edges[i][1]=y;
        adjmat[x-1][y-1]=1;
        adjmat[y-1][x-1]=1;
    }
    int sum,max=0;
    for (int i = 0; i < nvertex; i++)
    {
        sum=accumulate(adjmat[i].begin(),adjmat[i].end(),0);
        if(max<sum)
            max=sum;
    }
    max_degree=max;
    cout<<"Max Degree: "<<max_degree<<endl;
    inData.close();
    if(max_degree+1==nvertex)
    {
        cout<<"\nChromatic Number: "<<nvertex;
        exit(1);
    }
}
void Graph :: printAdjMat()
{
    cout<<"Adjacent Matrix:"<<endl;
    int c=1;
    for (int i = 0; i < nvertex+1; i++)
    {
        for (int j = 0; j < nvertex+1; j++)
        {
            if(i==0)
            {
                if(i==0 && j==0)
                    cout<<" \t";
                else
                    cout<<j<<"\t";
            }
            else
            {
                if(j==0)
                    cout<<i<<"\t";
                else
                    cout<<adjmat[i-1][j-1]<<"\t";
            }
        }
        cout<<endl;
    }
}

class ACOGCP
{
    public:
        string filename;
        vector<vector <bool>> adjmat;
        vector<int> color;
        int max_deg1,vertex,ncolor,edges;
        bool result;
        ACOGCP(Graph const &g1);
        bool check();
        int countDistinct();
        void compute();
        void run();
        void printColor();
};
ACOGCP::ACOGCP(Graph const &g1)
{
    filename=g1.filename;
    max_deg1=g1.max_degree+1;
    vertex=g1.nvertex;
    edges=g1.nedges;
    adjmat.resize(vertex,vector<bool>(vertex));
    color.resize(vertex,1);
    for(int i=0; i<vertex; i++)
    {
        for (int j = 0; j < vertex; j++)
        {
            adjmat[i][j]=g1.adjmat[i][j];
        }
    }
}
void ACOGCP::compute()
{
    int value, j, k;
    bool result;
    for (j = 1; j <vertex; j++)
    {
        ncolor=countDistinct();
        result=check();
        if (result==1)
        {
            return;
        }
        for (k = 0; k < vertex; k++)
        {
           if (j!=k)
           {
              value=adjmat[j][k]*color[k];
              if (value==color[j])
              {
                 if (color[k]<=max_deg1)
                    color[k]=color[k]+1;
                 else
                    color[k]=color[k]%(+max_deg1)+1;
                 j=0;
                 k=0;
              }
           }
        }
    }
}
int ACOGCP::countDistinct()
{
    unordered_set<int> s;
    int res = 0;
    for (int i = 0; i < vertex; i++)
    {
        if (s.find(color[i]) == s.end())
        {
           s.insert(color[i]);
           res++;
        }
    }
    return res;
}
bool ACOGCP::check()
{
    for(int i=0;i<(vertex-1);i++)
    {
        for (int j = i+1; j<vertex; j++)
        {
           if((color[i]==color[j]) && (adjmat[i][j]))
           {
             return 0;
           }
        }
    }
    return 1;
}
```

```
void ACOGCP::printColor()
{
   for (int j = 0; j < vertex; j++)
   {
      cout<<color[j]<<"\t";
   }
}
void ACOGCP::run()
{
   clock_t start, end;
   int a;
   cout<<"\n\nACOGCP:";
   start=clock();
   compute();
   result=check();
   end=clock();
   int chromatic_number=countDistinct();
   cout<<"\nChromatic Number: "<<chromatic_number<<endl;
   printColor();
   double time_taken = double(end - start)/double(CLOCKS_PER_SEC);
   cout << "\n\nExecution Time: " << fixed
        << time_taken << setprecision(5);
   cout << " sec " << endl;
   fstream fstream_ob;
   fstream_ob.open("output.csv", ios::app);
   fstream_ob<<filename<<","<<vertex<<","<<edges<<","<<chromatic_number<<","<<time_ta
ken<<endl;
   fstream_ob.close();
}
int main(int argc,char* argv[])
{
   try
   {
      if(argc==1)
         throw invalid_argument("No command line arguments specified. Filename not passed.");
      string file=argv[1];
      Graph g(file);
      //g.printAdjMat();
      ACOGCP aco(g);
      aco.run();
   }
   catch (const invalid_argument& e)
   {
      cout << "Error: " << e.what() << endl;
   }
}
```