gcolor: Graph Coloring

Jeffrey L. Duffany

September 9, 2009

1.0 Introduction

Given a graph G=(V,E) assign a positive integer to each of the vertices in such a way that no pair of vertices connected by an edge are assigned the same integer and the number of integers used is a minimum(k*). The graph G can be represented as an adjacency matrix A of ones and zeros. If A(i,j)=1 then there is an edge between nodes i and j. If A(i,j)=0 then there is no edge between nodes i and j. It is assumed that the matrix A is binary and symmetric and represents an undirected graph. In addition there can be no edge from a node to itself so that the main diagonal of A is zero. In other words A(i,i)=0 for i=1 to n where n is the number of vertices in the graph. The matrix A can also be recognized as representing a system of inequations[1]. The solution can be represented as a solution vector $s=(s_1,s_2,...s_n)$ where $s_i <= k$. If k=k* then s is an optimal solution.

2.0 The ineq Algorithm

The *ineq* algorithm[2][4] is used to find the solution vector s.

ineq(A)
i,j<-max(A^2)
if {i,j} = 0, return(A)
x_i<-x_i + x_j
A<- A - x_j
ineq(A)

Figure 1. The ineq algorithm

The algorithm works as follows. The solution vector s (not shown) is initialized to s=(1,2,...,n). The adjacency matrix A is squared and the maximum value of A^2 determined, ignoring any values corresponding to A(i,j)=1 since these vertices cannot be combined. Out of the set of all (i,j) such that A(i,j)=0 choose an (i,j) pair corresponding to the maximum value of A^2. Update the solution vector s to reflect the (i,j) pair combination (not shown). Take the ith row of A as a vector x_i and the jth row of A as vector x_j and perform the bitwise logical OR operation on the two vectors and assign the result to vector xi and column i and row i of A. Remove row j and column j from the matrix A and recursively call ineq(A) on the reduced A matrix. When there are no more remaining (i,j) pairs then the algorithm terminates, returning the original matrix A with the rows and columns permuted into the block diagonal form corresponding to the solution vector s (or the solution vector s itself).

3.0 Algorithm Performance

It can be shown that the ineq algorithm will find an optimal solution for all bipartite and complete k-partite systems and any system reasonably close to complete k-partite. In general there is no way to know that an optimal solution has been found unless k*=0,1,2 or 3. If ineq returns a solution vector s where max(s)=4 then it is possible that there could be a solution of k=3. However if ineq returns a solution vector s where max(s)=3 then it is certain that it is an optimal solution. The ineq algorithm can be made significantly faster by not squaring the A matrix on each iteration as shown in [5]. It is currently planned to provide the algorithm for this faster version in a future update.

Many of the instances that are not solved by ineq can be solved by one of the more powerful variations of ineq. One variation [3] extends the decision function A^2 to a power series of A^n and uses a gradient search of the coefficients to find an optimal solution. Another variation[6] uses a gradient search on the number of zeros in the block diagonal, combining sets of variables to transform the system into another which is in the region of convergence of the decision function where it can be solved. It is currently planned to provide the algorithms for these more powerful versions in a future update to this package.

4.0 Using the gcolor Package

The graph G=(V,E) must first be represented as a binary symmetric matrix A with all zeros on the main diagonal. For convenience, a function test(n,k) has been provided where n is the size of the matrix and k is the cardinality of the optimal solution. The test program picks a random ones density for the A matrix where the ones density is the number of 1 elements in the A matrix divided by the maximum number of ones (n^2-n). The test(n,k) code can be modified to set this to some fixed value or to otherwise modify the range of allowed values for the ones density. Alternatively you can import the A matrix from outside of R or create A within R using the matrix command. The following example generates a 10x10 test matrix with optimal solution cardinality k*=3 then finds a solution using ineq:

```
a<-test(10,3)
ineq(a)

[1] 1 2 1 3 2 1 1 1 2 2
```

The output of the ineq command is the solution vector s of integers corresponding to the coloring of the vertices of the graph G. The solution vector will always represent a valid coloring although it may or may not be optimal. In general the only way to know if the solution is optimal (max(s)=k*) is to create the test matrix A with a known optimal solution cardinality k* as can be done with the function test(n,k). If ineq provides a solution vector s such that max(s) = k* then you know that an optimal solution has been found.

Functions are also provided to import the adjacency (A) matrix from outside of R in either uncompressed of compressed DIMACS format[7]. The function for uncompressed DIMACS format is importDIMACSAscii() and the function for compressed DIMACS format is importDIMACSBin(). A DIMACS format file can be downloaded from a website such as [7] where the .col file extension is used to indicate an uncompressed DIMACS file and the .col.b extension is used to indicate compressed DIMACS. A file can be imported into R as follows:

```
fpsol21<-importDIMACSAscii("fpsol2.i.1.col")
flat1000_50<-importDIMACSBin("flat1000_50_0.col.b")
```

depending on whether it is compressed or not. If in either of the above cases the file name is not specified as a function parameter then a popup window will appear so that the user can browse the directory structure for the desired file.

5.0 Acknowledgements

The importDIMACSAscii and importDIMACSBin functions were written by Euripides Rivera Negron who also did all of the work in creating and assembling all of the files necessary to submit the color package to CRAN.

6.0 References

1. Wikipedia – Inequation page: http://en.wikipedia.org /wiki/Inequation

2. Duffany, J.L. “Systems of Inequations”, 4th LACCEI Conference, Mayaguez, PR, June 21-23, 2006.

3. Duffany, J.L. “Generalized Decision Function and Gradient Search Technique for NP-Complete Problems”, XXXII CLEI Conference, Santiago Chile, August 20-23, 2006.

4. Duffany, J.L., “Statistical Characterization of NP-Complete Problems”, Foundations of Computer Science Conference, World Computer Congress, Las Vegas, Nevada, July 14-17, 2008.

5. Duffany, Jeffrey, “Optimal Solution of Constraint Satisfaction Problems", International Conference on Applied Computer Science, Sharm el Sheik, Egypt, December 29-31, 2008.

6. Duffany, Jeffrey, “Equivalence Class Subset Algorithm”, International Conference on Computer and Information Technology”, Tokyo, Japan, May 27-29, 2009.

7. http://mat.gsia.cmu.edu/COLOR/instances.html