Chromnum: A software to compute the chromatic number[1] of a graph implementing a novel trailing path algorithm.

Area: Classical Graph Theory (Mathematics), NP-Hard Problems in Computer Science
Language: MATLAB, Octave
Status: Unpublished
Authors: Abhirup Bandyopadhyay[1] & Sankar Basu*[2]

Highest Academic Degree:
[1] M.Tech in Mathematics (Specialization: Operation Research)
[2] Ph.D in Biochemistry (Specialization: Computational Biophysics)
* Corresponding author

Email: Sankar Basu (nemo8130@gmail.com)
Abhirup Bandyopadhyay (abhirupnit@gmail.com)

Requires: Octave[2] or MATLAB (version R2016a or higher)

Platform specifications
chromnum_octave.m runs on MATLAB as well as on Octave while chromnum.m runs only on MATLAB (v. R2016a or higher).

Run Commands

run the script chromnum.m or chromnum_octave.m in MATLAB command window
or
chromnum_octave.m in Octave
Both scripts act as functions with a single input argument which is the filename (with full path or locally called from the current directory) containing the adjacency matrix.

- Chromnum computes the chromatic number of a graph
- Chromnum implements a novel **trailing path** meta-heuristic approximation algorithm (see Glossary).
- The function takes the 1-0 adjacency matrix stored in a text file as its sole input argument
- And returns the chromatic number of the corresponding graph

Usage:
(Default)

```
crn = chromnum('adj.inp')
crn = chromnum_octave('adj.inp')
```

Additional Features:

- The program also generates a color map for the graph and tabulates the same, if asked to.
- Please note that this color map could be degenerate and may alter between two successive runs of the program on the same graph.

---

1 The chromatic number is the minimum number of distinct colors that is required to color or label the nodes of a graph so that no two adjacent nodes share the same color.

2 Noncommercial analog of MATLAB, used mainly in Linux platform

- The chromatic number in most cases should be identical between the degenerate colormaps or else the minimum value obtained over a finite number of iterations is to be considered the best approximated solution.
- Furthermore, the program also returns the order at which the nodes have been colored. Again this may have degenerate solutions.
- One can store the colormap and the order by putting two more output variables while running the program.
  i.e.,

  Usage:
  (Advanced)

  [crn, cmap] = chromnum_octave('adj.inp')
  [crn, cmap, order] = chromnum_octave('adj.inp')

Example input file content:

- Each row should contain the adjacencies of one node in the graph
- The matrix entries should be separated by a single white-space

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

OUTPUT:
(Default)

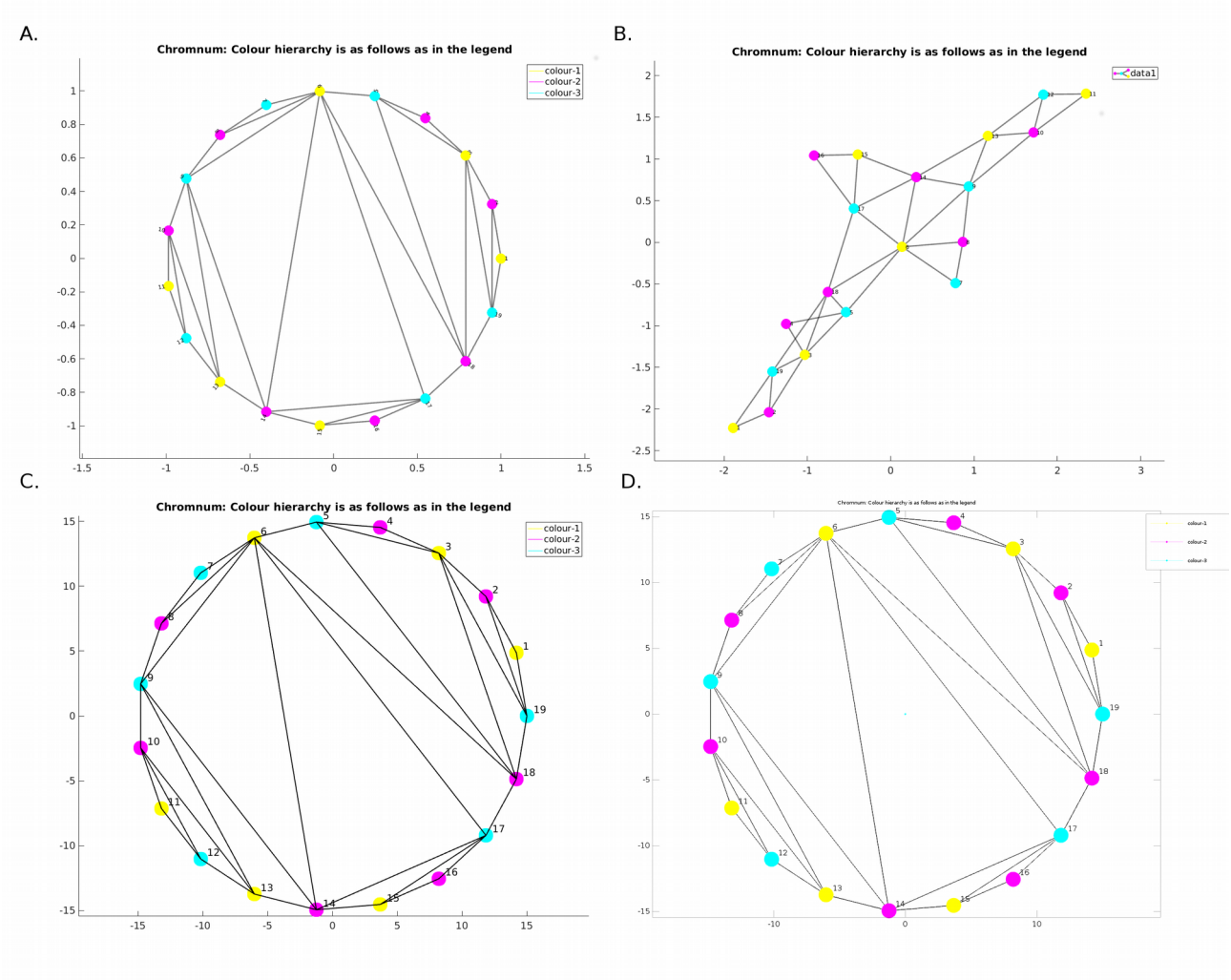
Number of Nodes in the given undirected graph : 5
Number of edges : 7
Chromatic Number : 3

Proposed Colormap:

Node-1 : color-3
Node-2 : color-2
Node-3 : color-2
Node-4 : color-3
Node-5 : color-1

Graphical Output

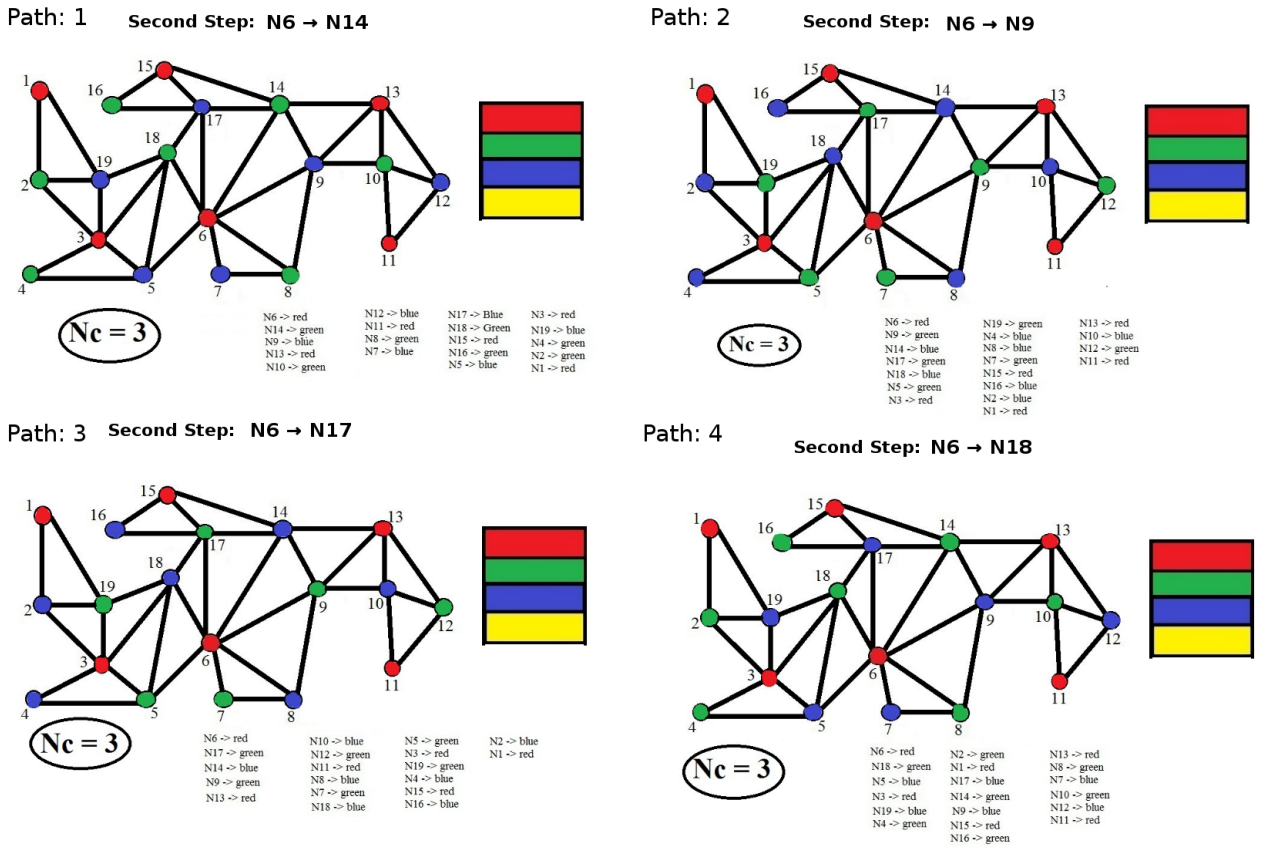
The graphical outputs generated by different versions of the program on the same network. There are two versions of the program chromnum.m and chromnum_octave.m both runs on MATLAB while the later is made compatible with Octave. chromnum.m generates visual outputs in both circular (shown in Panel: A) and forced (Panel: B) layout in MATLAB. The graphical outputs generated by chromnum_octave.m is also shown in Panel C (in MATLAB) and Panel D (Octave) – both in circular layout. The network is 3-colorable and the identical one used to demonstrate the algorithm.



Glossary

Demonstration of the trailing path algorithm.

The example is shown as a demonstration of the trailing path algorithm. The algorithm leads to four possible degenerate paths in its second iterative step, subsequent to coloring the *highest-degree-node* (N6 → red) in the first step. These four paths are: N6 → N14 (path 1), N6 → N9 (path 2), N6 → N17 (path 3) and N6 → N18 (path 4). Note that all of the possible four nodes have identical degrees of five (which is the highest degree among all neighbors connected to N6) and identical lengths of their color array (19-1=18). In four independent runs covering all four degenerate paths, the algorithm returns the same chromatic number of three while exploring two possible alternative colormaps (say cm1 and cm2). Note that the colormap assigned to path1 and path4 are the same (cm1) while the ones assigned to path2 and path3 are also identical (cm2).



The Trailing Path Algorithm

Let G=(V, e) be a graph with N nodes. Let A_i represent an array, referred as the 'color array' for the i^{th} node, which stores all available colors that can be used to color it, i.e. all those colors by which any neighbor of the i^{th} node is not yet labeled. The initial length of the color array, A_i is then set to N; as the minimum number of colors to label N nodes must be lesser than or equal to N. Thus, initially A_i is set to an ordered array of N colors for each (i^{th}) node in the network. The algorithm has two hierarchical levels in its structure. In the first level, the algorithm starts coloring nodes from the *highest-degree-node*. In cases of degenerate paths where there exist more than one nodes with the highest degree, it starts coloring from an arbitrarily chosen *highest-degree-node* and assigns colors to nodes along different random paths from this *highest-degree-node*. In each iteration the algorithm trails through different random paths and labels each i^{th} node on the path sequentially by

the first available color in the corresponding color array A_i. At each iteration, subsequent to coloring the i^{th} node, all edges incident to that (i^{th}) node are deleted, along with the node itself. Thus, at each step, the algorithm encounters a new and unique induced subgraph of the original graph. In case of disjoint subgraphs, the algorithm colors them separately, independent of each other. In the next (second) level, the algorithm starts its trailing path from the node which has the least number of colors available for labeling it. Likewise to the earlier level, the algorithm arbitrarily chooses a node in cases of degenerate paths, that is to say that, if there exist more than one nodes with the same least number of available colors. By this way, the algorithm keeps assigning labels to a graph iteratively by the trailing path and keeps track of the minimum number of colors required to label all nodes till convergence. Finally, this updated least number of colors is returned as the value of the chromatic number. The structure of the trailing path algorithm could be written as follows.

1. Repeat the whole process N times for different random trailing paths in the graph, until the chromatic number converges.
    1.1. Do while: highest degree of the graph is positive.
    Find the highest degree node q.
    Start coloring by the first available color in the color array A_q.
    Delete the color from the color array of all the nodes connected to node q.
        1.1.1. Do while: q has at least one neighbor
        Chose a neighbor p of q with the least number of colors in the color array.
        Delete all the edges which are incident to q.
        Assign a color to p by the first available color in the color array A_p.
        Delete the color from the color array of all the nodes connected to node p.
        If p has a neighbor r with degree 1, assign r the first available color from the corresponding color array, A_r.
        Delete all the edges which are incident to p.
        Reset q=p.
    1.2. Find the chromatic number as the minimum number of colors used to color the graph in each iteration.
2. Return the convergent value of the chromatic number.

To note is that the algorithm involves randomness from the very first step of the algorithm. It starts coloring nodes from any *highest-degree-node* taken at random (in case of more than one *highest-degree-nodes*). It then chooses a path from that *highest-degree-node* to a neighboring node which has the minimum number of available colors (or the maximum number of unavailable colors[i]) to label it. In case of a tie between two or more neighboring nodes in this parameter, it chooses a node which has a degree higher than that of all other neighboring nodes. If there's another tie in terms of the degree also, it chooses a path randomly. The salient feature of moving through only connected nodes makes the algorithm follow a 'trailing path'. Again, the involvement of randomness in every nontrivial step imparts an evolutionary attribute to the metaheuristic where the corresponding number of colors determines the fitness of a 'trailing path'.

Plausible Area of Application:

1. Any physical mapping problem like the maps defining geographical or state territories
2. Partitioning of Structured Networks in any field
3. Security control in an art galary by a collection of surveillance camera
4. Regulating the production of explosive gaseous substances in a chemical hub
5. Protein design in structural biology

---

i For any finite color array