# Greedy Graph Coloring and Parallel FEM Assembly

Tomás Zegard

UIUC

October 25, 2010

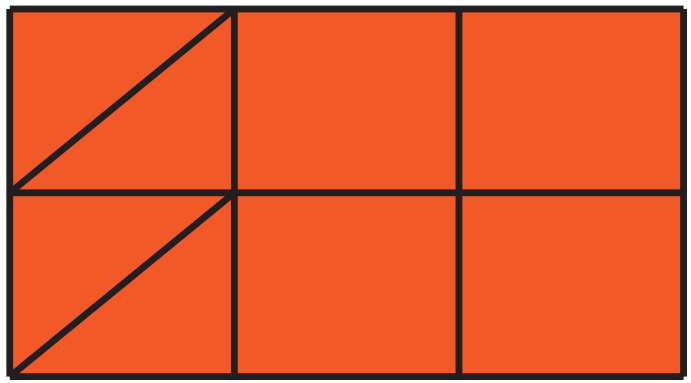
## Goal

The assembly of stiffness matrices in FEM suffers from race condition if two adjacent elements are assembled at the same time. Using graph coloring we prevent this situation from ever happening.

- What is graph coloring?
  Graph coloring consists of coloring a graph (or map, or mesh) in such a way that no two adjacent nodes (or countries, or elements) share the same color.
- Why graph coloring?
  If we assemble the system matrix one color at a time, two adjacent elements will never get assembled concurrently effectively eliminating the race condition.
  Keep in mind that graph coloring is not the only technique that solves this problem.

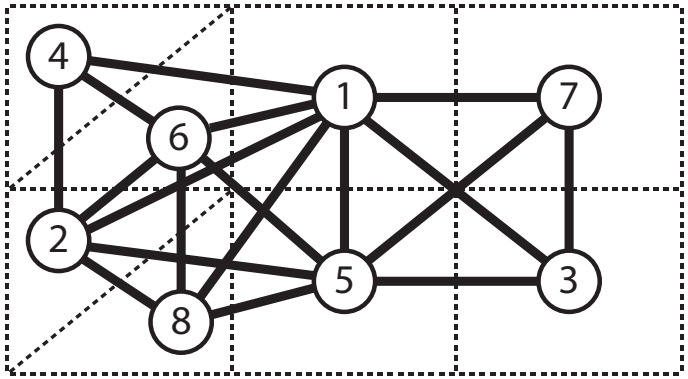
# Communication or Adjacency Matrix

- In order to do the coloring, we require to know the neighbors for each element. This information is contained in the *communication matrix* or *communication graph*. The former is easier to understand, and the latter is just another way to store the same information.
- For the following mesh



if we number all elements, we can then obtain the connectivity graph $G$ as in the next image

# Communication or Adjacency Matrix




from there the communication matrix $L\left(G^C\right)$ of for the graph $G$ is:

$$L\left(G^C\right) = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

# Greedy Coloring Algorithm

where $L_{i,j} = 1$ if elements $i$ and $j$ share one or more nodes (are neighbors), and 0 otherwise, and $L_{i,i} = 0$ by definition (keep in mind that some other communication matrix definitions may consider $L_{i,i} = 1$).

## Greedy Coloring Algorithm

1. Get the next element in the mesh
2. Traverse all neighbors using $L\left(G^C\right)$, and see what colors cannot be used
3. Color this element with the next available color
4. If this is not the last element, goto 1

This algorithm is good for unconstrained coloring (as in a world map). But for parallel computing, we have a limited number of resources (or threads). Hence **each color cannot be used more than** a certain number of times, usually equal to **the number of available threads**.

# Greedy Coloring Algorithm

An algorithm that considers this situation would be

## Greedy Coloring Algorithm w/ thread restriction

1. Get the next element in the mesh
2. Traverse all neighbors using $L\left(G^C\right)$, and see what colors cannot be used
3. Pick the next available color
4. If color is already full, goto 3
5. Color this element with this color
6. If this is not the last element, goto 1

Let's walk through an example...

# Coloring Example

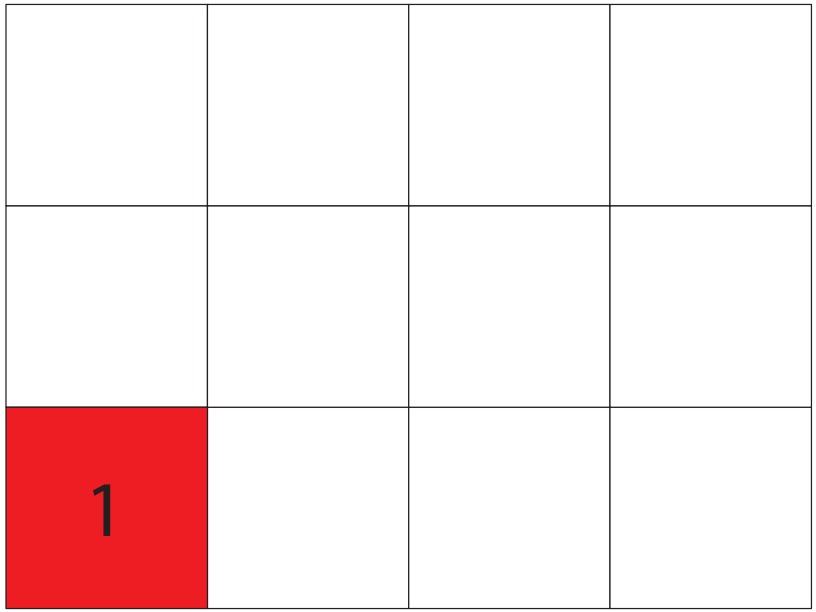
The mesh to be colored is a structured $4 \times 3$ mesh of quads. We commence by numbering every element as follows

| | | | |
|---|---|---|---|
| 9 | 10 | 11 | 12 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |

The thread restriction will be 3. This means that **no color can have more than 3 elements**.

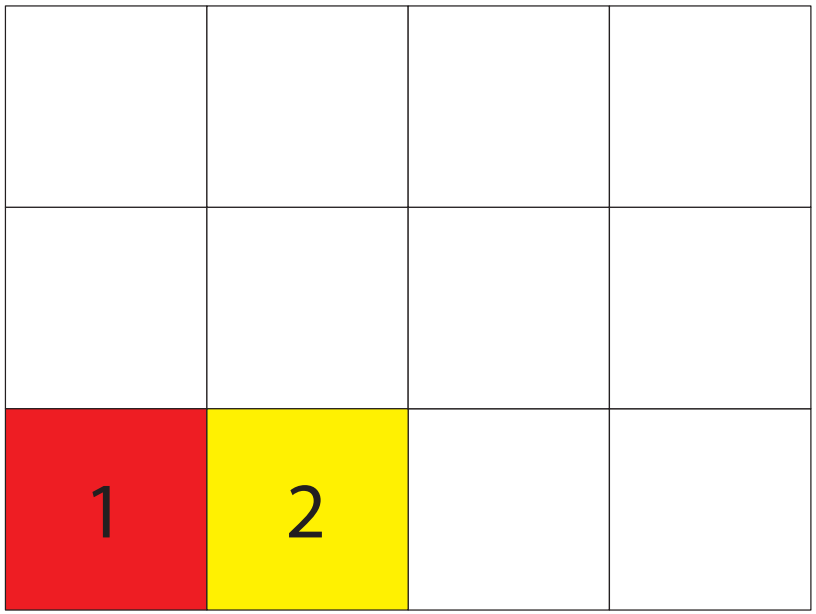
# Coloring Example

We commence by coloring the first element with color 1.



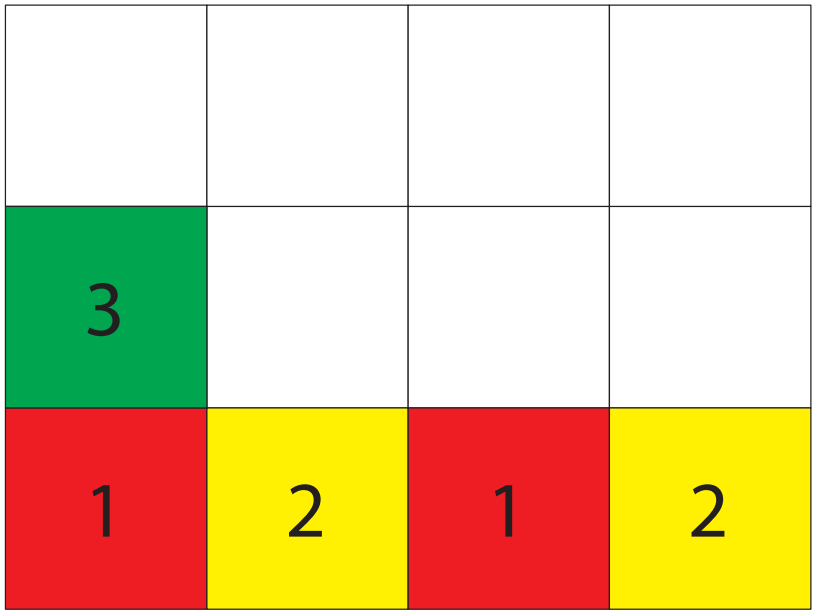
# Coloring Example

Traversing the neighbors of the second element, that is the columns with value of 1 of row 2 in $L\left(G^C\right)$, we discover that we cannot use color 1. Since color 1 is all we have, a new color is created. Hence element 2 is colored with color 2.



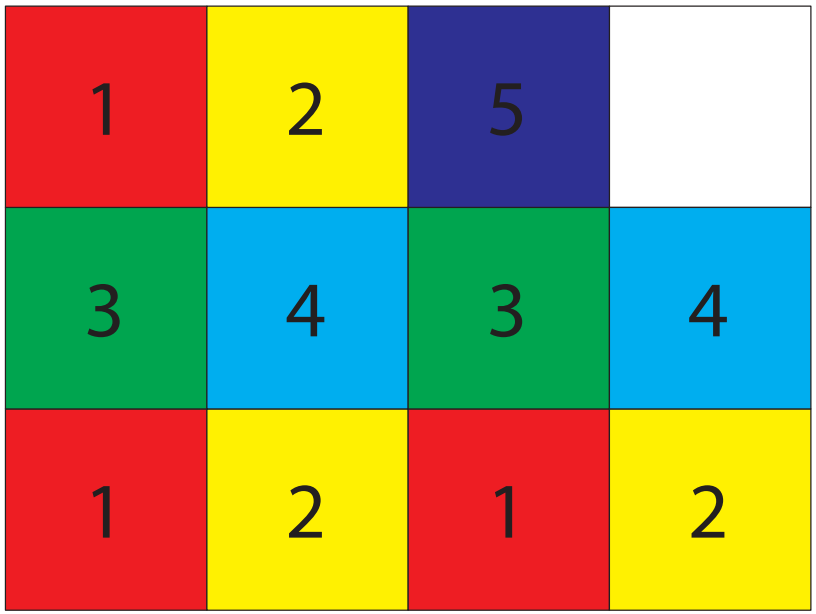
# Coloring Example

Continuing the same procedure, we have to create color 3 at element 5 because we cannot use color 1 nor 2 because they are in use by the neighbors.



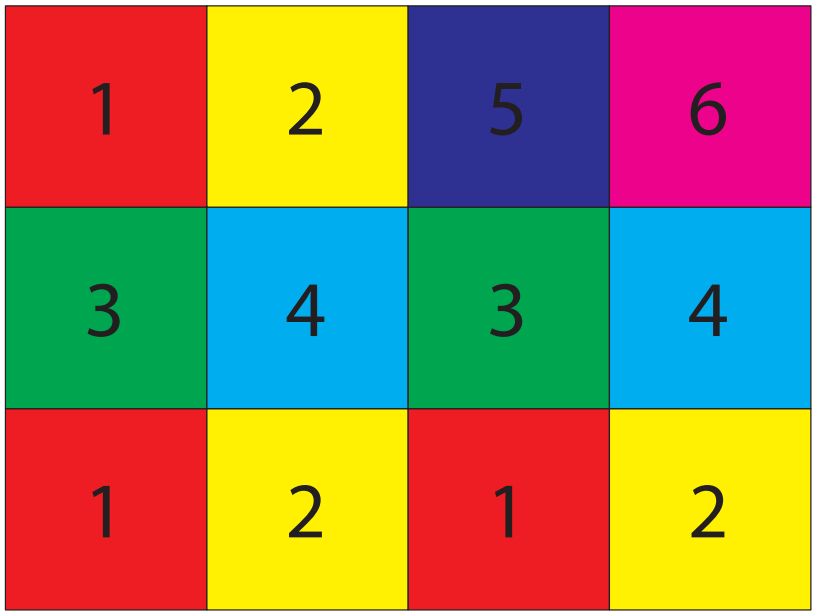
# Coloring Example

A very interesting situation happens at element 11. Technically color 1 is available for this element, but because the color is full (the thread limitation is 3), a new color has to be created.



# Coloring Example

The final coloring for this mesh is then




this coloring uses 6 colors if the number of available threads is 3.

# Final Remarks

Now all that is left is to assemble all the elements from each color at a time step before moving to a new color. This solves the race condition encountered in most parallel FEM codes.

Final words:

- The greedy algorithm is simple, yet very fast for coloring graphs
- A MATLAB® code with the algorithm and more complicated examples is given
  Run the file *RunExamples.m* to see other examples
- We encourage you to experiment with the greedy coloring algorithm and have fun...
  Suggestion: Analyze the efficiency of the algorithm keeping the thread limitation fixed and increasing the mesh size.

# References


- A. H. Gebremedhin, F. Manne and A. Pothen - What Color Is Your Jacobian? Graph Coloring for Computing Derivatives - *SIAM Review, Vol. 47, No. 4, pp. 629-705 (2005)*
- C. Cecka, A. J. Lew and E. Darve - Assembly of finite element methods on graphic processors - *International Journal for Numerical Methods in Engineering (2010)*
- A. Unterkircher and J. Reissner - Parallel assembling and equation solving via graph algorithms with an application to the FE simulation of metal extrusion processes - *Computers and Structures, Vol. 83, Issue 8-9 (2005)*
- R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt - Concepts and Applications of Finite Element Analysis - *Wiley, 4 edition (2001)*