Finding Maximal Cliques Using MATLAB

Dr Richard Mitchell

Cybernetics Intelligence Research Group

Department of Cybernetics

The University of Reading, UK

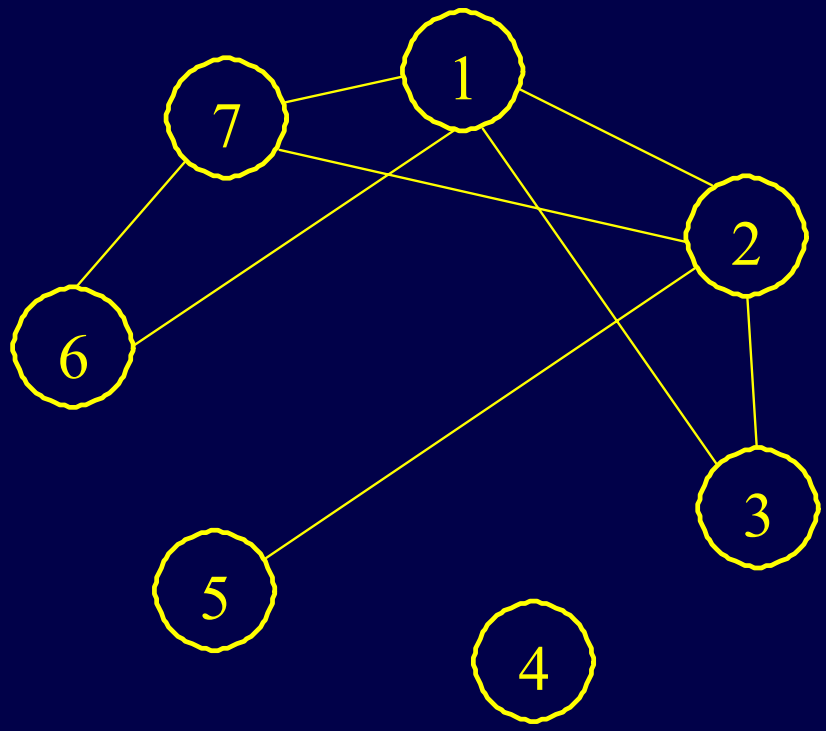
R.J.Mitchell@reading.ac.uk

Work done during Sabbatical at  National Grid

Abstract

- Maximal Clique Problem
  - well known NP complete problem
- Early good algorithm: Bron Kerbosch
  - Backtracking + Sets to stop repeats
  - Written in obscure Algol 60
  - Uses nodes with most connections
- What if use nodes with least connections
- Both implemented in MATLAB
- Show hybrid of two may be best

Maximal Clique Problem



Have nodes, some are connected

Fully connected nodes are cliques

Can group 1,2,7; 1,2,3; 1,6,7 and 2,5

Seems easy, but is NP complete problem

Applications of Cliques

- In Mean Tracking Cluster Algorithm
  - Windows move through data space
  - Those which overlap are merged
  - Find all pairs to merge
  - May be mutual pairs to merge
- Merging in async seq logic states
  - Combine all states mutually mergable
- Graph theory
  - Graph has nodes connected by edges
  - Find all fully connected sets of nodes

Bron Kerbosch

- Backtracking plus Candidates & Nots Sets and Compsub – has nodes in next clique
- Extend(Nots, Candidates, Compsub)
  - Find node with most connections
  - for number of non connections

```
Select node
Nots = Nots connected to node
Candidates = Candidates connected to node
If any Candidates
   Extend (Nots, Candidates, Compsub+node)
Else Compsub is next clique
Move Node from Candidates to Nots
```

New Algorithm

- Backtrack but select node with fewest connections
- Process node then remove
- Why? NP gets worse with N
  - So work on small problems
  - Main problem becomes smaller
- Works ok to certain extent
- To stop repeat searches, have ‘nots’
- But this is list of nots rather than set

On Implementation

- Standard backtrack on 785 pairs
- Took 26,320 seconds in MATLAB
- RJM algorithm first attempt: 2.62 secs!
- Bron Kerbosch MATLAB similar
- Careful coding in MATLAB
  - use built in MATRIX functions
  - (less code to interpret)
  - Work on columns not rows
  - Algorithm took 0.2 secs

Testing

- ought to do algorithm O() analysis
- Easier, get MATLAB to do run and time it
- Test on graphs with
  - 10,15,20, … 50 nodes
  - Each with 10%, 30%, 50%, 75%, 90% or 95% interconnected

10 nodes @30%: 14 edges, 11 cliques

40 nodes@75%: 546 edges, 1816 cliques

50 nodes@90%:1103 edges, 119778 c's

Times for Bron Kerbosch

	10%	30%	50%	75%	90%	95%
10	0	0.0020	0.0060	0.0040	0.0040	0.0040
15	0.0020	0.0060	0.0100	0.0140	0.0040	0.0080
20	0.0060	0.0100	0.0240	0.0300	0.0320	0.0160
25	0.0060	0.0240	0.0380	0.1040	0.1200	0.0520
30	0.0100	0.0340	0.0740	0.2060	0.5270	0.3140
35	0.0180	0.0440	0.1240	0.4050	0.9790	0.5330
40	0.0220	0.0620	0.1940	0.7990	3.4930	3.1450
45	0.0280	0.0920	0.3020	1.5100	12.354	7.1180
50	0.0380	0.1100	0.4150	2.1290	55.450	30.820

RJM algorithm

	10%	30%	50%	75%	90%	95%
10	0	0.0040	0.0040	0.0040	0.0020	0
15	0.0020	0.0080	0.0120	0.0160	0.0040	0.0060
20	0.0040	0.0180	0.0400	0.0440	0.0360	0.0100
25	0.0080	0.0360	0.0660	0.1940	0.1660	0.0500
30	0.0180	0.0600	0.1380	0.3560	0.9810	0.4290
35	0.0260	0.0900	0.2400	0.7670	2.0070	0.9230
40	0.0420	0.1260	0.3950	1.6760	7.8010	5.7140
45	0.0660	0.1820	0.5890	3.2450	30.1410	15.0640
50	0.0840	0.2340	0.9010	4.5290	157.807	88.5450

Conclusion and Further Work

- Usually Bron Kerbosch better
- For highly connected, RJM can be better
- RJM spends time searching ‘not list’
- Hybrid algorithm perhaps worthwhile

- Consider better ‘not list’
- Do formal algorithm order analysis.