Community detection

Continuous assessment activity #3

Lorenzo Venieri - April 2022

Brief description of the algorithms and the programs used:
For community detection I used algorithms from Igraph and NetworkX.
In particular I used from Igraph:
- *community_fastgreedy*: based on greedy optimization of modularity, I also tried to use *community_optimal_modularity* that does an extensive search to find the partition that gives the highest modularity but for the big networks this computation takes a very big time.
- *community_label_propagation*: label propagation algorithm by Raghavan, Albert and Kumara that gives nodes an initial label that is propagated to neighbors nodes (by majority). It is not deterministic.
- *community_leading_eigenvector:* optimization of modularity using the eigenvectors of the modularity matrix.

I tried also two of these algorithms from NetworkX:
- *greedy_modularity_communities*: analogous to *community_fastgreedy* from Igraph
- *label_propagation_communities*: analogous to *community_label_propagation* from Igraph
- *k_clique_communities*: I tried also the k-clique algorithm but decided to not include it in the results since it requires to tune the parameter k for each network. It can be found in my notebook commented.

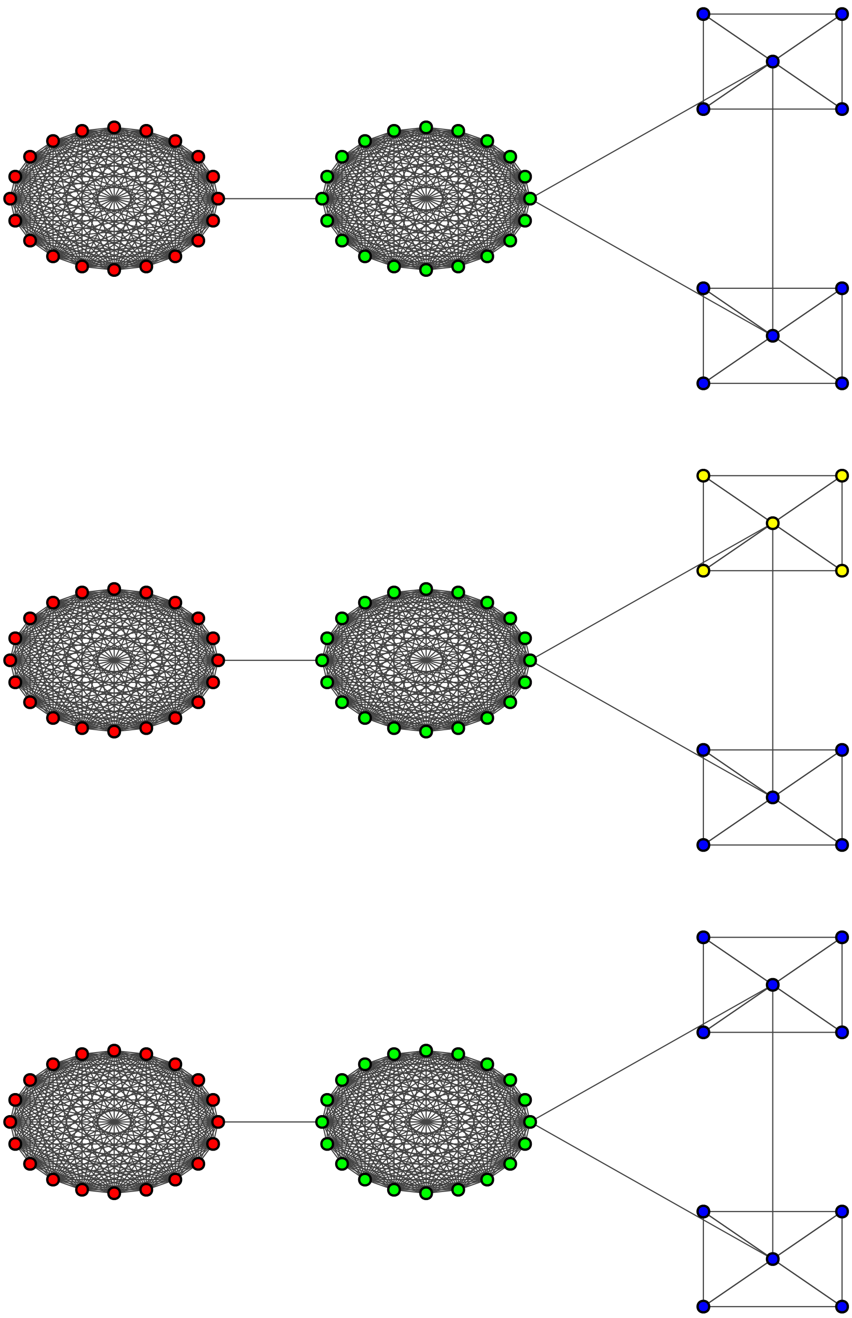
The scripts used can all be found in the notebook attached as a pdf (CommunityDetection_lvenieri.pdf)

Plots with color-coded communities:

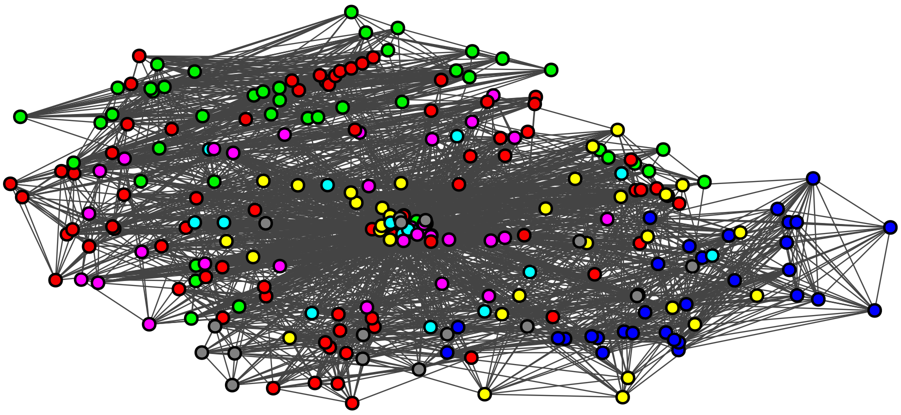
In the next pages are shown the plots of the graph color-coded by community. They will be ordered by algorithm used for the detection of the communities:

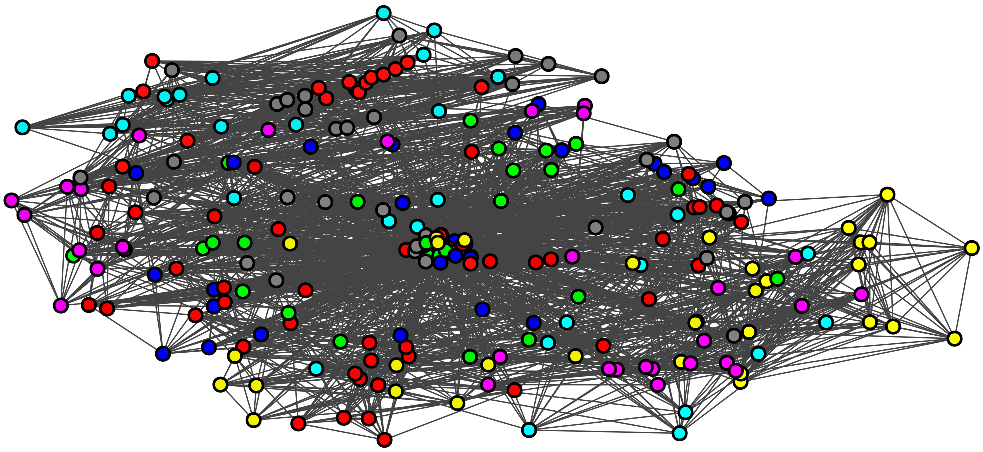
1) *fastgreedy*
2) *label propagation*
3) *leading eigenvector*

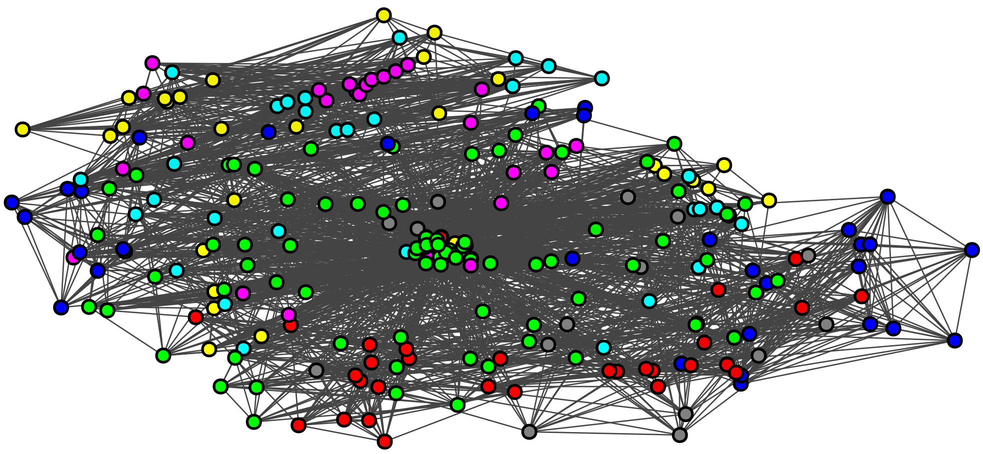
Graph: 20x2+5x2



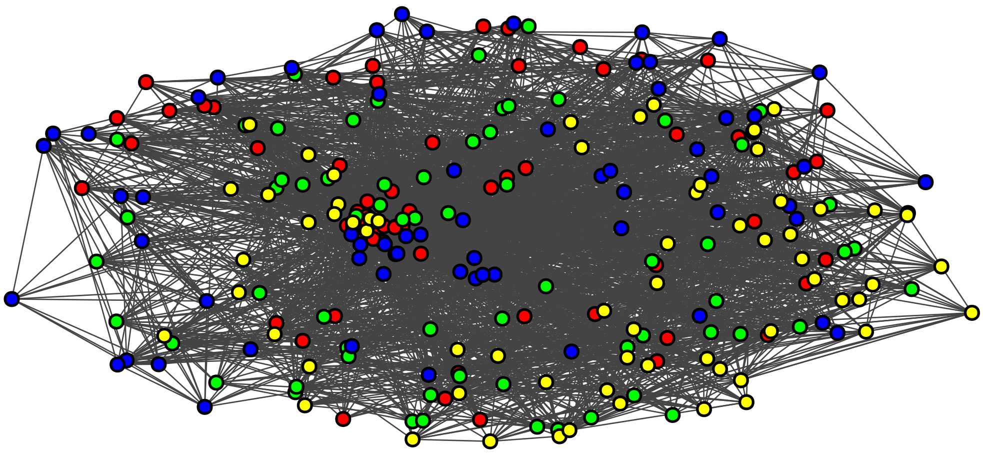
Graph: 256_4_4_2_15_18_p

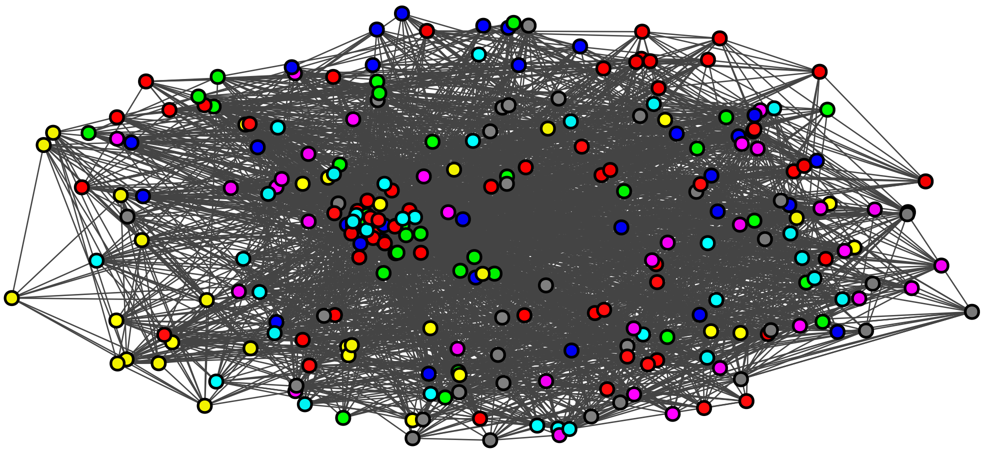


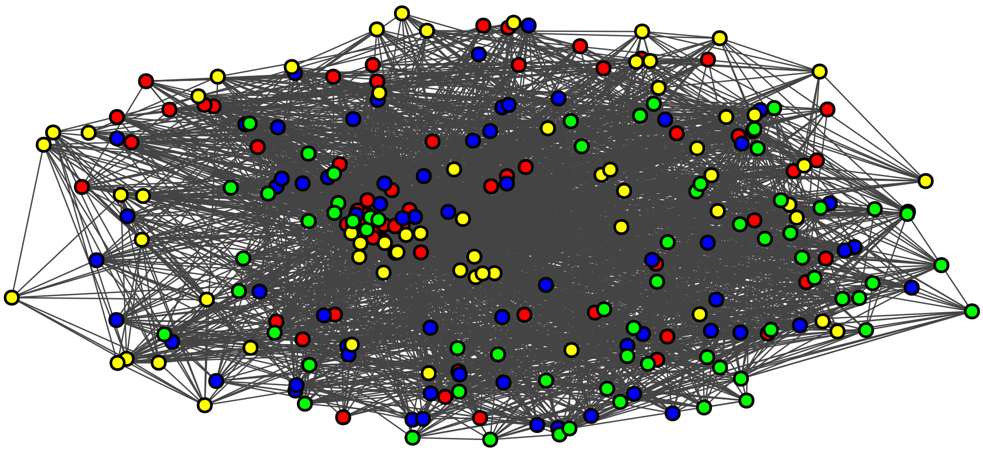




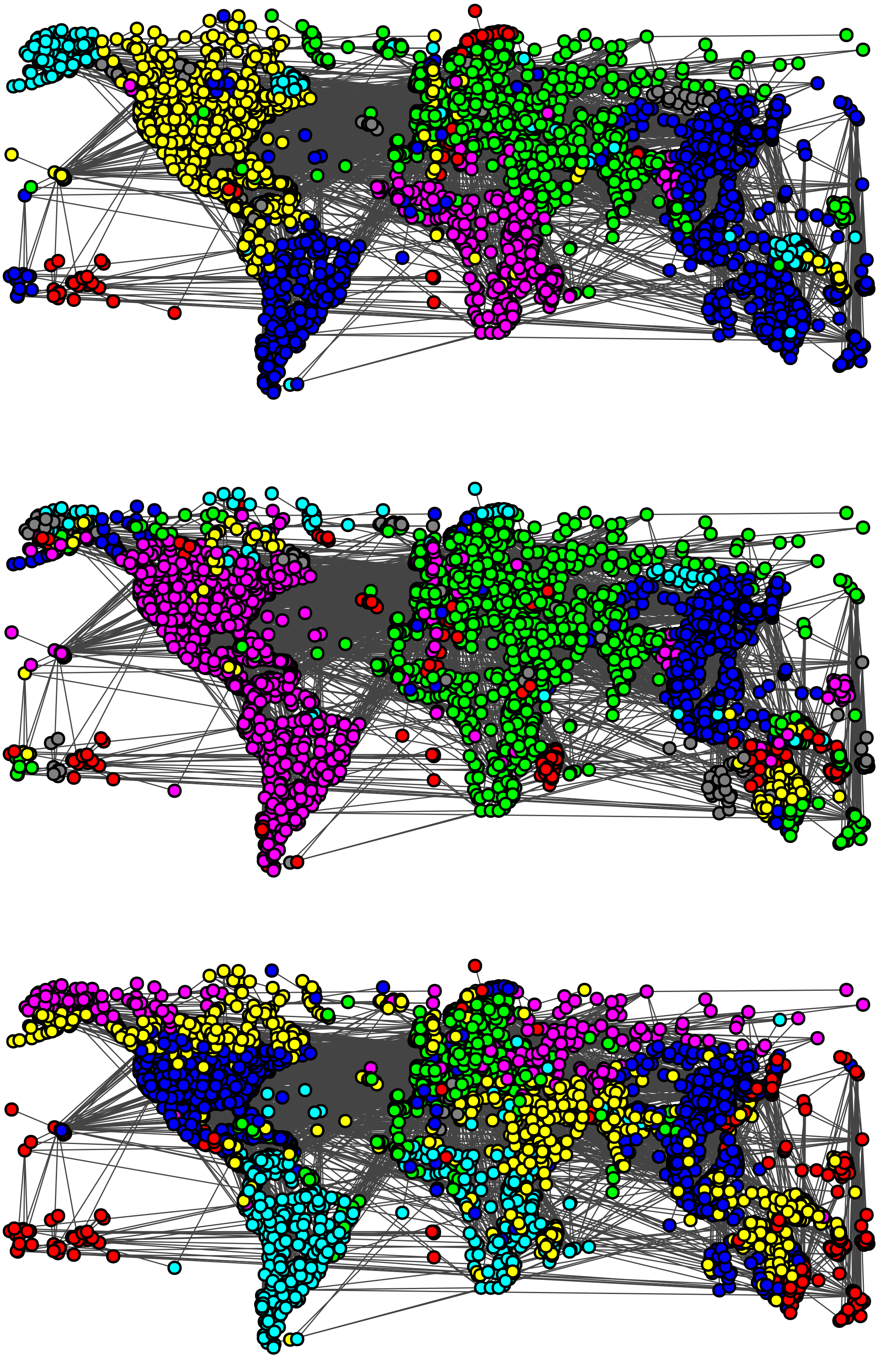
Graph: 256_4_4_4_13_18_p



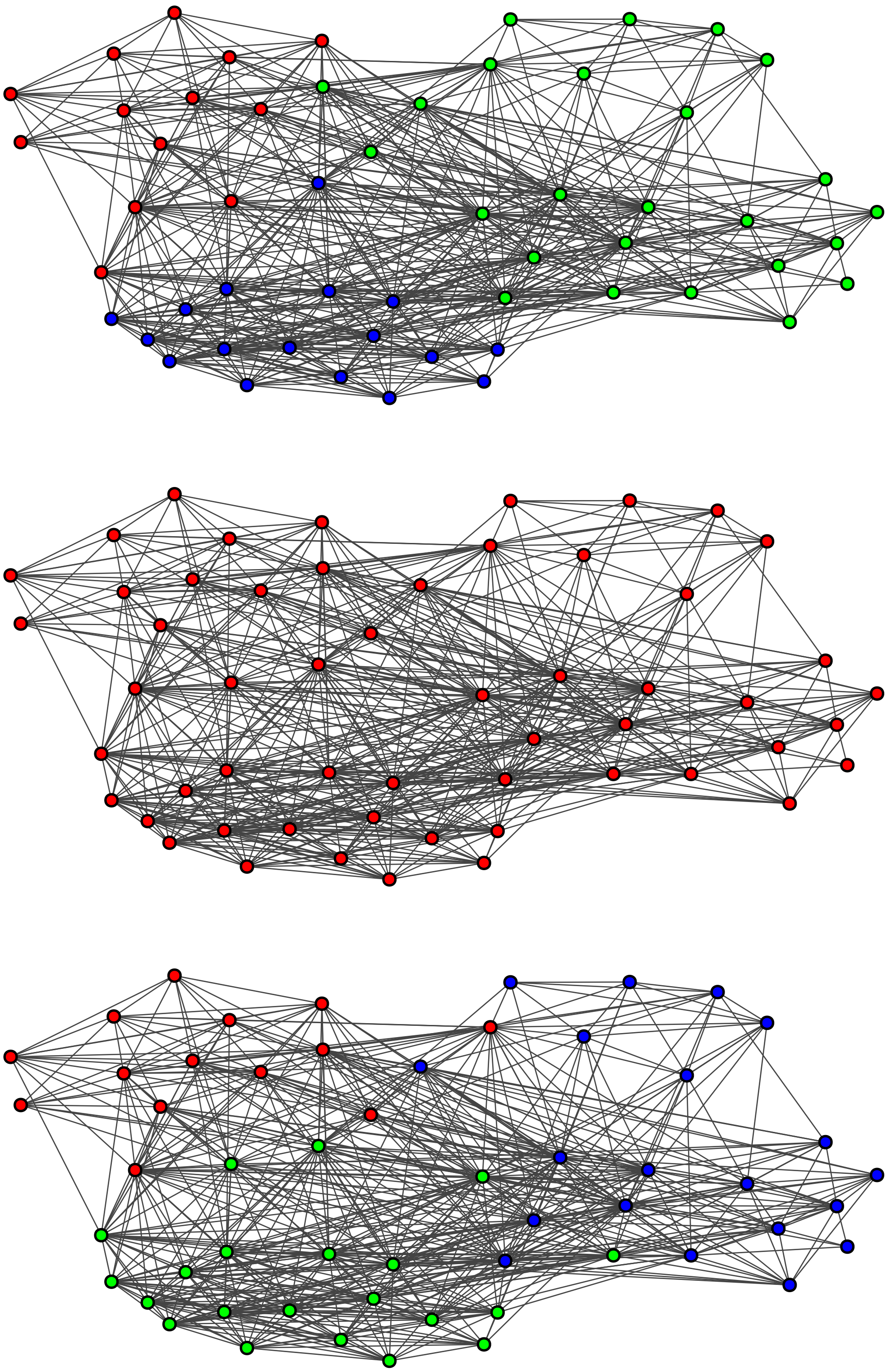




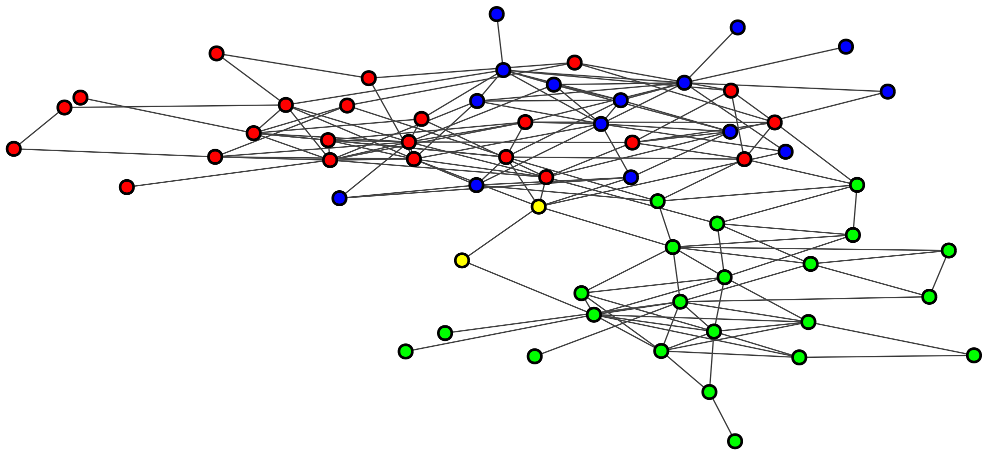
Graph: Airports_UW

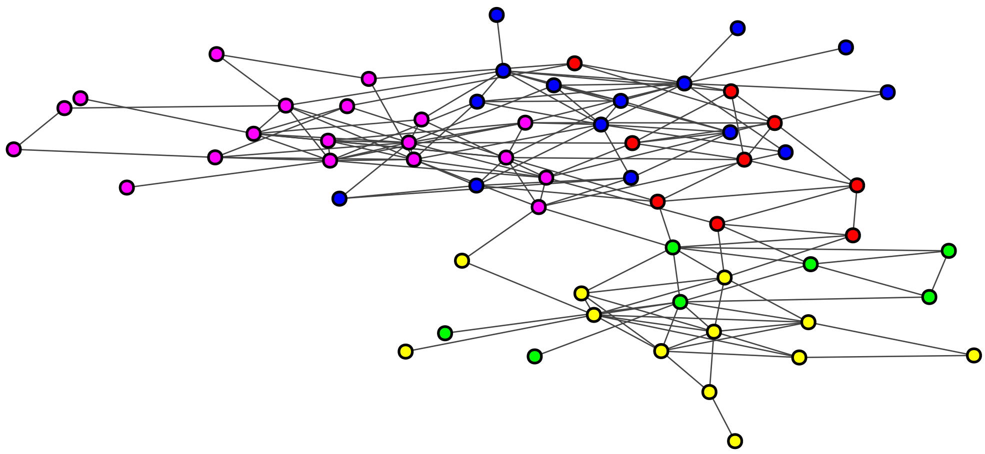


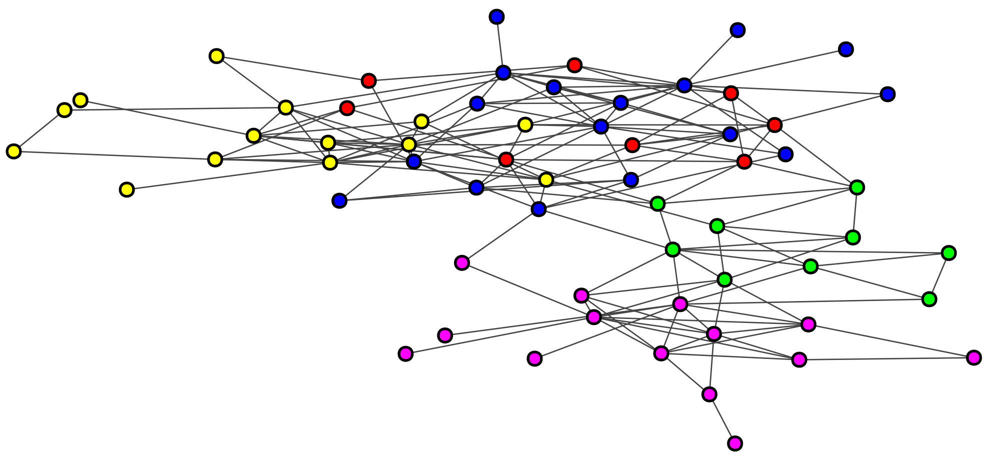
Graph: cat_cortex_sim



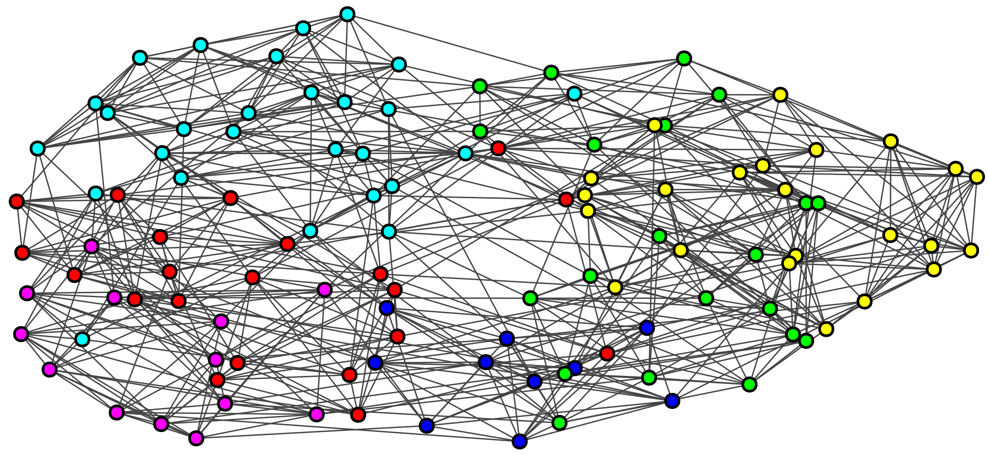
Graph: dolphins

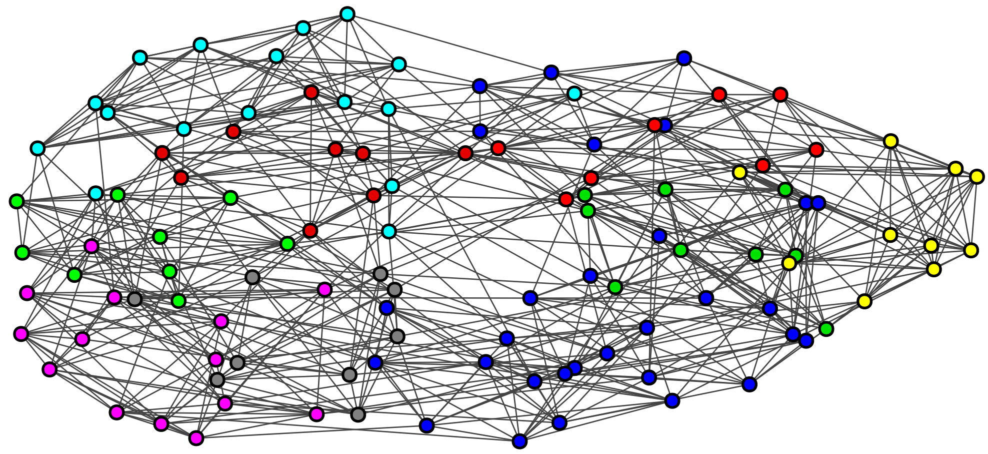


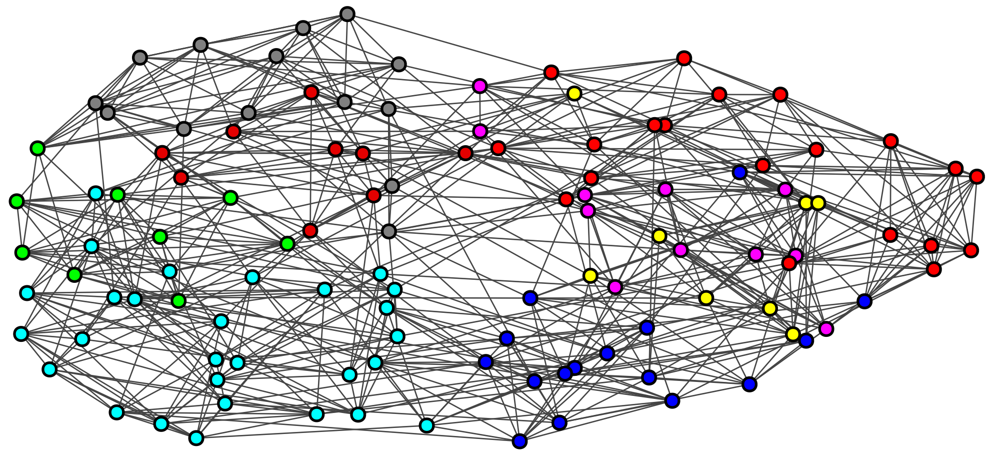




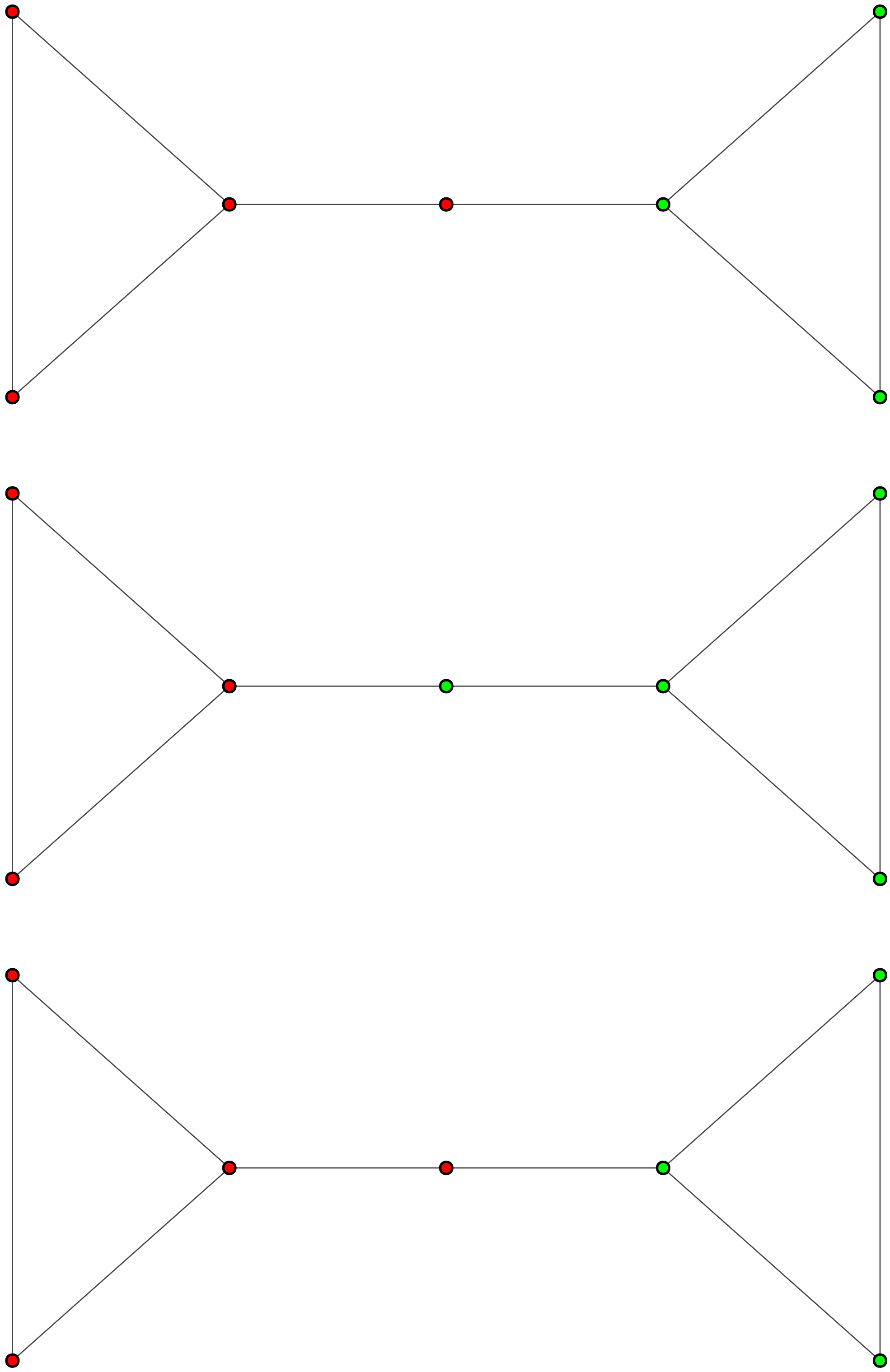
Graph: football





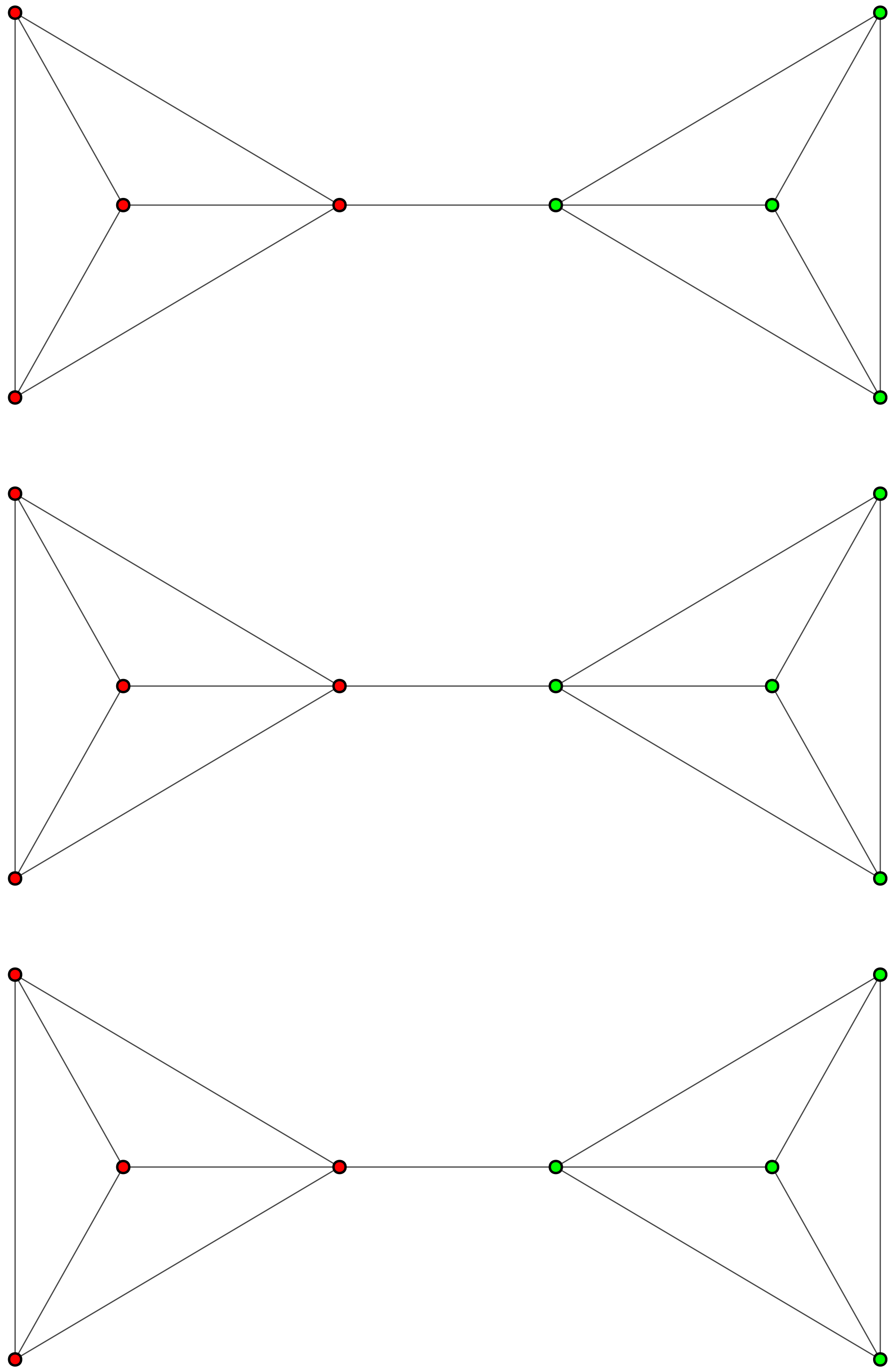


Graph: graph3+1+3



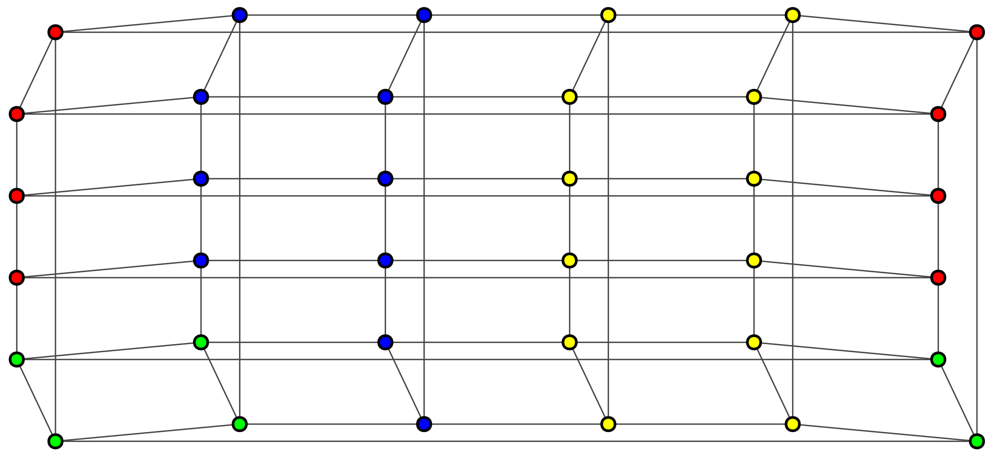
For this graph we can see that the only node that has an ambiguous membership to one of the two communities is the node in the center since it has one edge that connects it to each community.

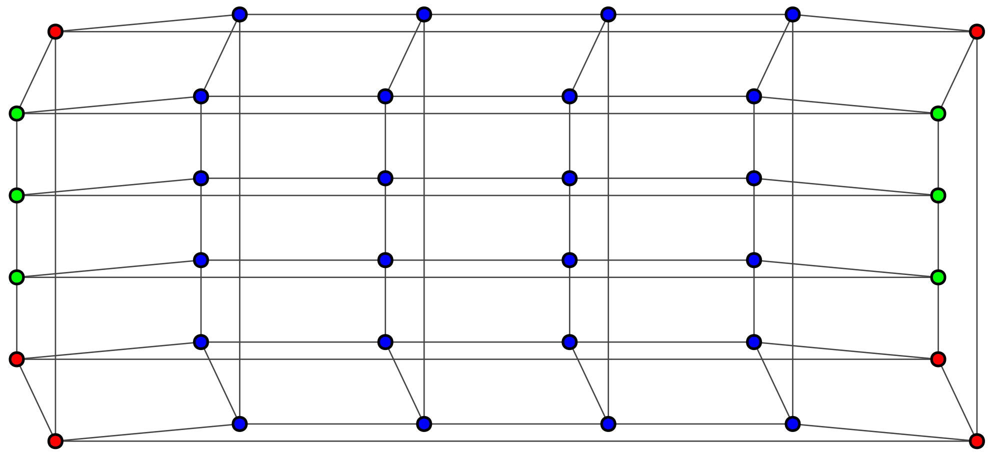
Graph: graph4+4

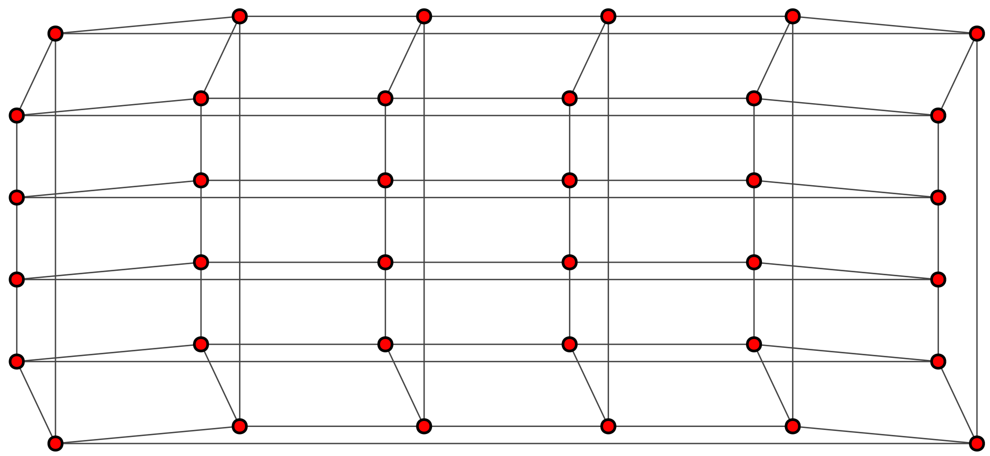


For this graph the partition into communities is obviously not ambiguous since there are two distinct cliques linked by a single edge.

Graph: grid-p-6x6

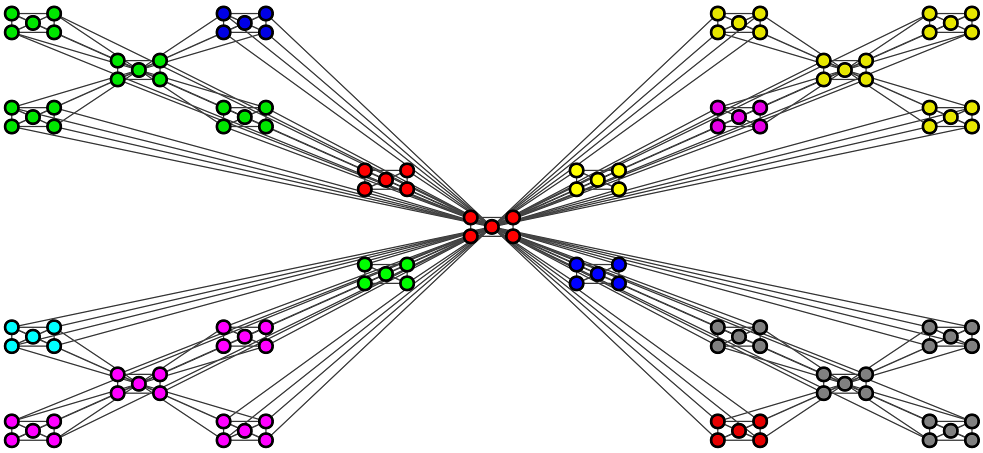


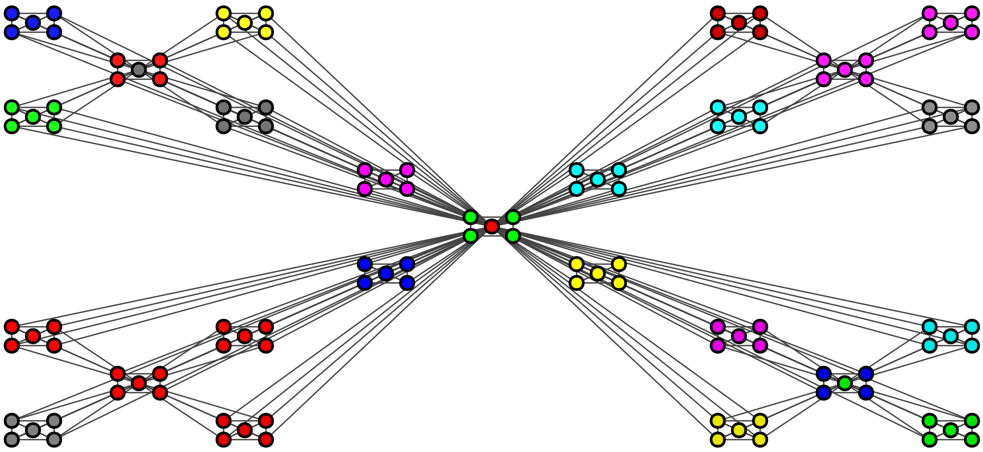


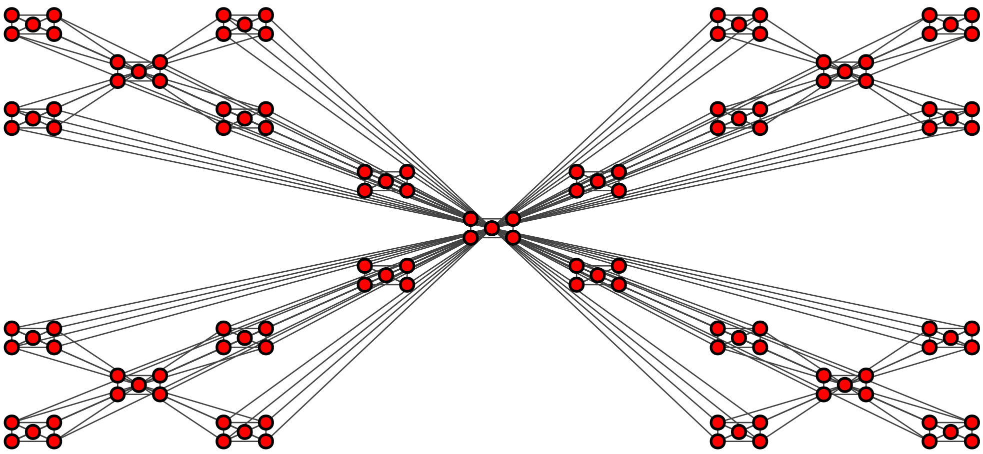


In this graph the partition has to be random since each node has the same exact properties of the others: exactly 4 neighbors.

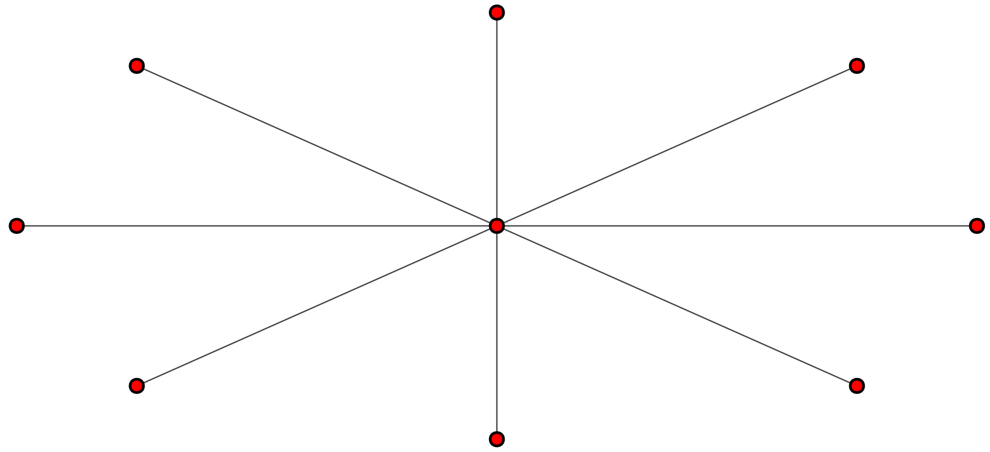
Graph: rb125

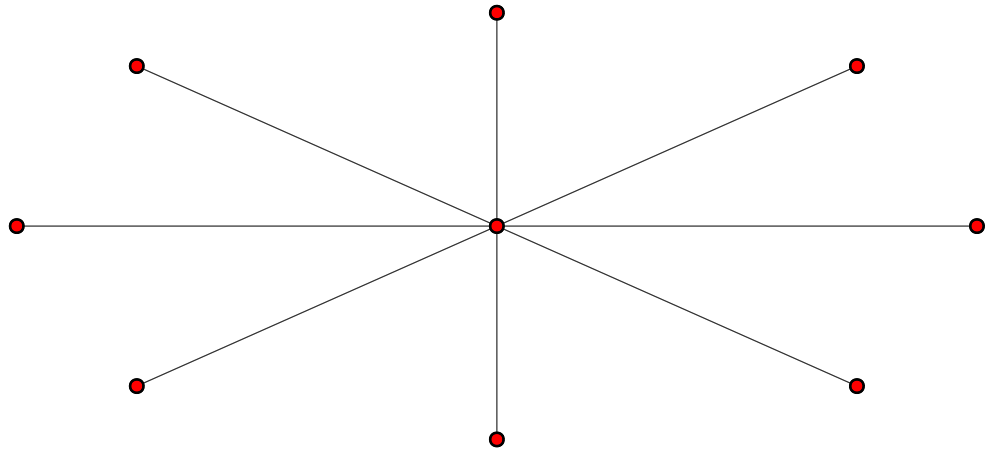


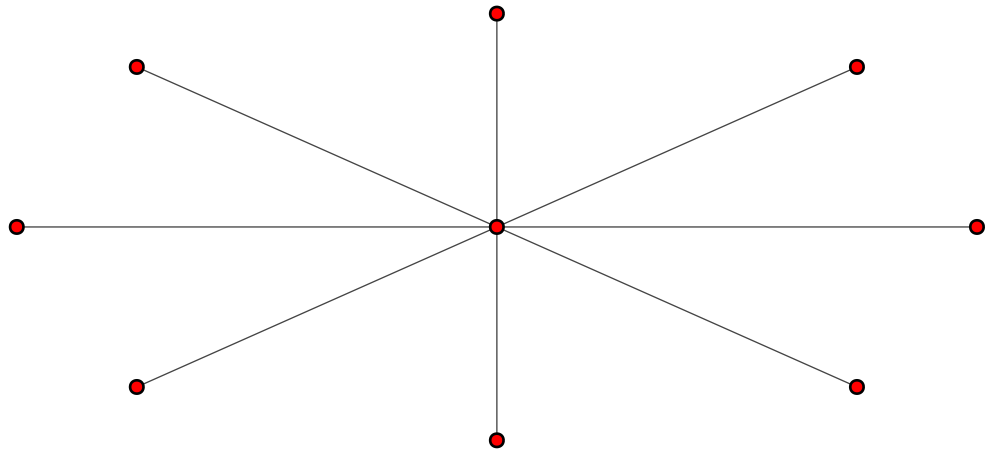




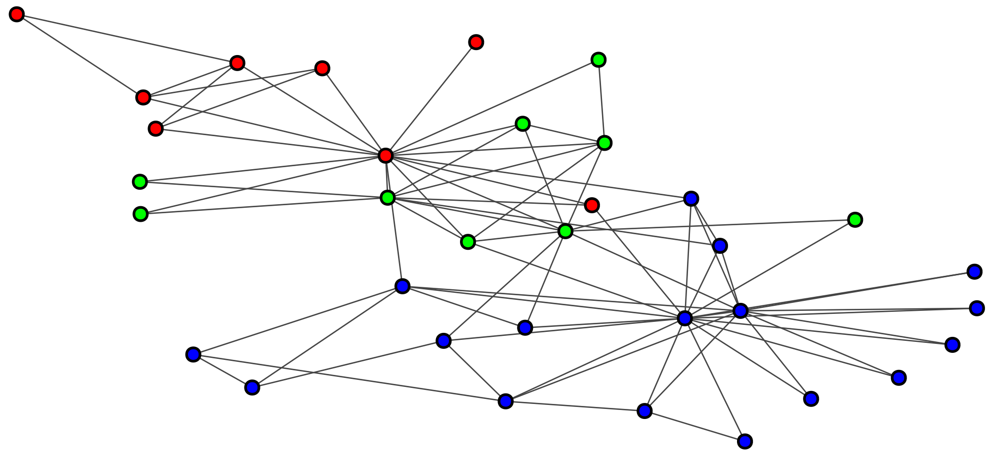
Graph: star

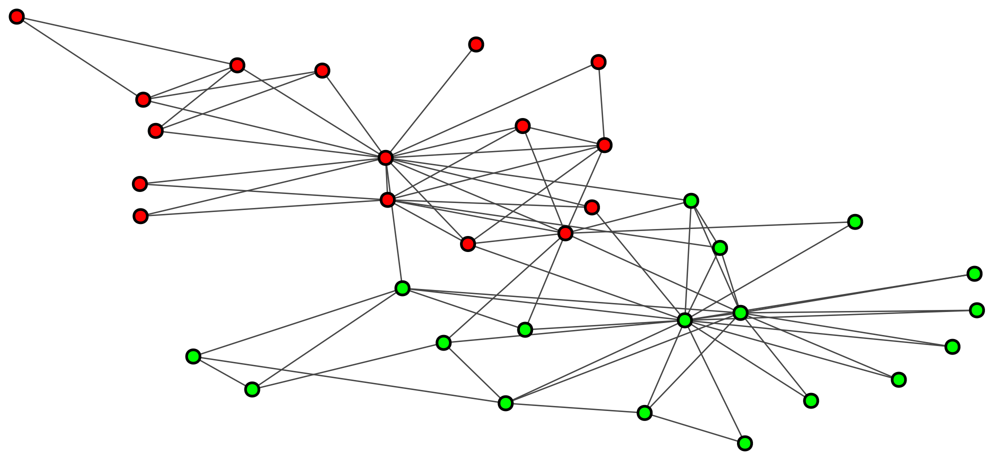


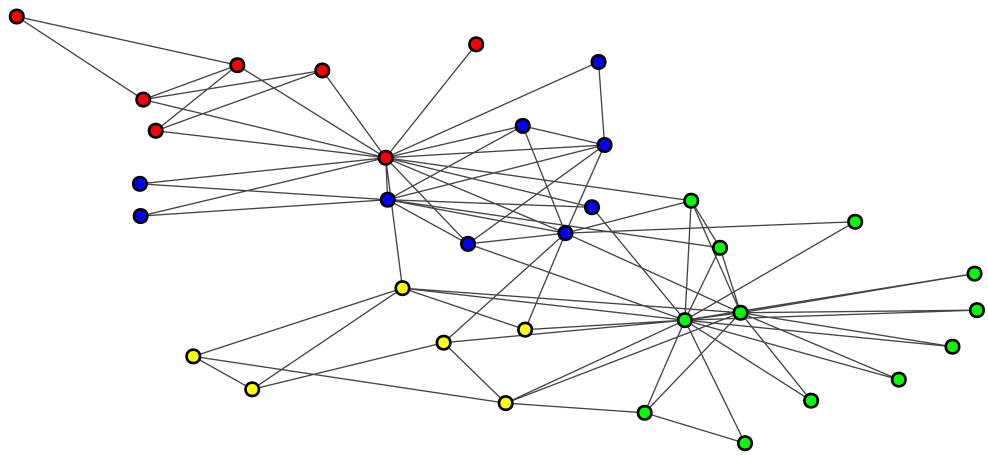




Graph: zachary_unwh







Comparison measures between partitions found and reference ones:

Partitions found with Igraph:

20x2+5x2

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.941176	0.938345	0.051124
1	leading_eigenvector	0.941176	0.938345	0.051124
2	label_propagation	1.000000	1.000000	0.000000

256_4_4_2_15_18_p

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.483871	0.869708	0.166303
1	leading_eigenvector	0.542431	0.924071	0.102676
2	label_propagation	1.000000	1.000000	0.000000

256_4_4_4_13_18_p

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	1.000000	1.000000	0.000000
1	leading_eigenvector	1.000000	1.000000	0.000000
2	label_propagation	0.269841	0.680851	0.338132

cat_cortex_sim

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.542169	0.656873	0.296602
1	leading_eigenvector	0.547872	0.618651	0.332598
2	label_propagation	0.257239	0.000000	0.481994

rb125

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.281143	0.825443	0.287653
1	leading_eigenvector	0.031742	0.000000	0.967777
2	label_propagation	0.324100	0.887932	0.198671

star

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	1.000000	1.000000	0.000000
1	leading_eigenvector	1.000000	1.000000	0.000000
2	label_propagation	1.000000	1.000000	0.000000

dolphins

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.504125	0.572700	0.271613
1	leading_eigenvector	0.329314	0.448914	0.423804
2	label_propagation	0.943044	0.888836	0.049311

football

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.362153	0.697732	0.385871
1	leading_eigenvector	0.350324	0.698670	0.407185
2	label_propagation	0.545143	0.848038	0.210807

graph3+1+3

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.666667	0.809540	0.238237
1	leading_eigenvector	0.666667	0.809540	0.238237
2	label_propagation	0.666667	0.809540	0.238237

graph4+4

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	1.000000	1.000000	0.000000
1	leading_eigenvector	1.000000	1.000000	0.000000
2	label_propagation	1.000000	1.000000	0.000000

zachary_unwh

	algorithm	Jaccard	NMI	NVI
0	fastgreedy	0.683274	0.692467	0.217697
1	leading_eigenvector	0.505495	0.677092	0.269804
2	label_propagation	0.462845	0.225967	0.351184

Partitions found with Networkx:

20x2+5x2

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.941176	0.938345	0.051124
2	label_propagation	1.000000	1.000000	0.000000

dolphins

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.245068	0.010911	0.628715
2	label_propagation	0.203363	0.041141	0.717362

256_4_4_2_15_18_p

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.461653	0.845852	0.196748
2	label_propagation	0.245902	0.773341	0.266577

football

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.058675	0.147975	1.086858
2	label_propagation	0.044610	0.238707	1.113253

256_4_4_4_13_18_p

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	1.000000	1.000000	0.000000
2	label_propagation	0.413333	0.699300	0.263123

graph3+1+3

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.666667	0.809540	0.238237
2	label_propagation	0.666667	0.809540	0.238237

cat_cortex_sim

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.571930	0.702341	0.257299
2	label_propagation	0.257239	0.000000	0.481994

graph4+4

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	1.000000	1.000000	0.000000
2	label_propagation	1.000000	1.000000	0.000000

rb125

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.281143	0.825443	0.287653
2	label_propagation	0.159132	0.754033	0.420127

zachary_unwh

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	0.379009	0.289122	0.503219
2	label_propagation	0.403361	0.254134	0.494754

star

	algorithm	Jaccard	NMI	NVI
0	greedy_modularity	1.000000	1.000000	0.000000
2	label_propagation	1.000000	1.000000	0.000000

Modularity values:

	partitions found		
	fastgreedy	leading_eig	label_prop
graph3+1+3	0.367188	0.367188	0.367188
20x2+5x2	0.542579	0.542579	0.541586
graph4+4	0.423077	0.423077	0.423077
grid-p-6x6	0.401235	0.000000	0.277778
star	0.000000	0.000000	0.000000
cat_cortex_sim	0.260436	0.255355	0.000000
zachary_unwh	0.380671	0.393409	0.364809
dolphins	0.495491	0.491199	0.392073
airports_UW	0.662490	0.639231	0.516750
football	0.549741	0.492606	0.540977
256_4_4_4_13_18_p	0.696773	0.696773	0.664728
256_4_4_2_15_18_p	0.765660	0.752151	0.781804
rb125	0.608733	0.000000	0.583748

	reference
	modularity
graph3+1+3	0.351562
20x2+5x2	0.541586
graph4+4	0.423077
star	0.000000
cat_cortex_sim	0.245996
zachary_unwh-real	0.371466
dolphins-real	0.373482
football-conferences	0.553973
256_4_4_4_13_18_p	0.696773
rb125-1	0.600595
rb125-2	0.558144
rb125-3	0.553147
256_4_4_2_15_18_p	0.781804

Conclusions:
Looking to the comparison measures between the partitions found and the reference ones I'd say that the algorithm that most often gives more similar results to the reference partitions is the label propagation algorithm.
This of course does not apply to every graph analyzed: for instance for the graph 256_4_4_4_13_18_p we have very low scores for the metrics of similarity with the reference partition (and obviously a very high Normalized Variation of Information). It has to be taken into account that the label propagation algorithm is not deterministic so running again the algorithm with the same graph may give different results.
Analyzing the modularity tables it is obvious that we usually have bigger modularity scores for the partitions found by the modularity optimizing algorithms. This is not necessarily a measure of the fact that the label propagation algorithm is performing worse since modularity is just one of the quality measures to quantitatively say how good a given partition into communities is, and there is no universally accepted definition.