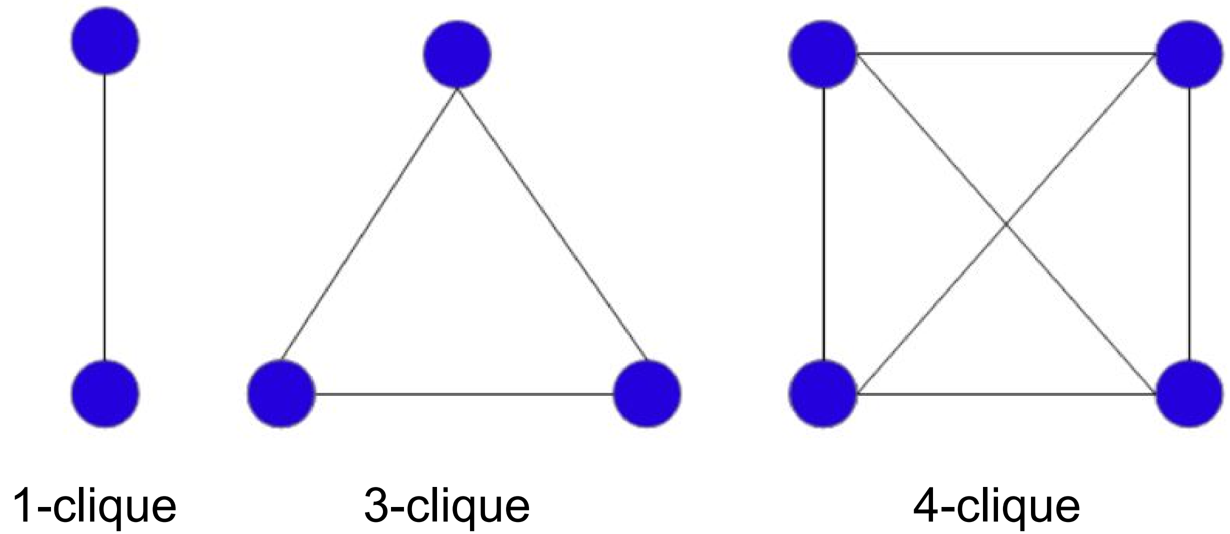
The k clique community finding algorithm

Alexis Gallèpe
Ignace Agbogba
Marina Leão Lucena

Description of the algorithm

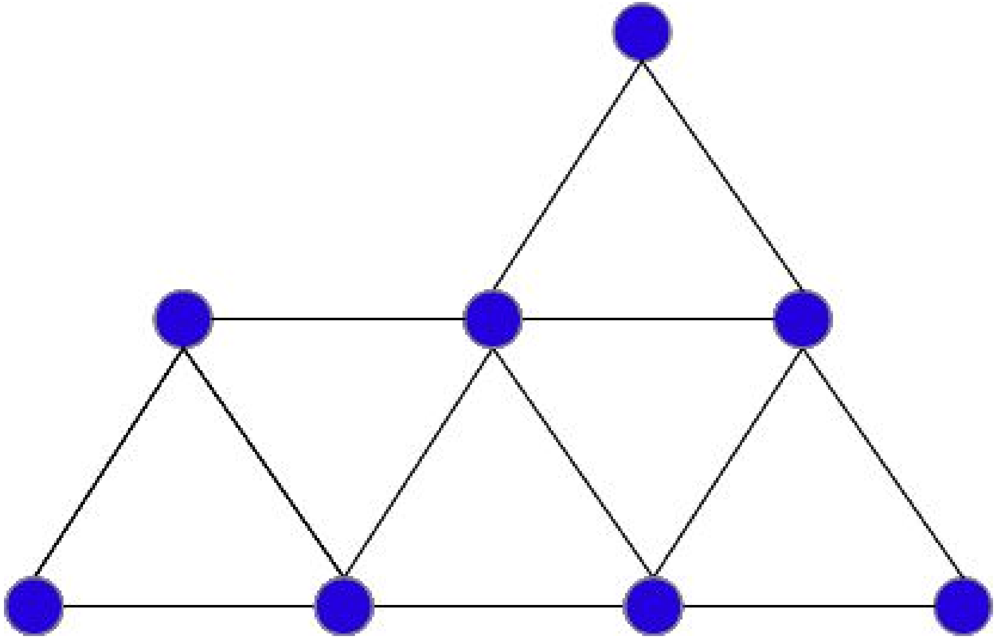
What are k-cliques?



Description of the algorithm

What is a k-clique community?

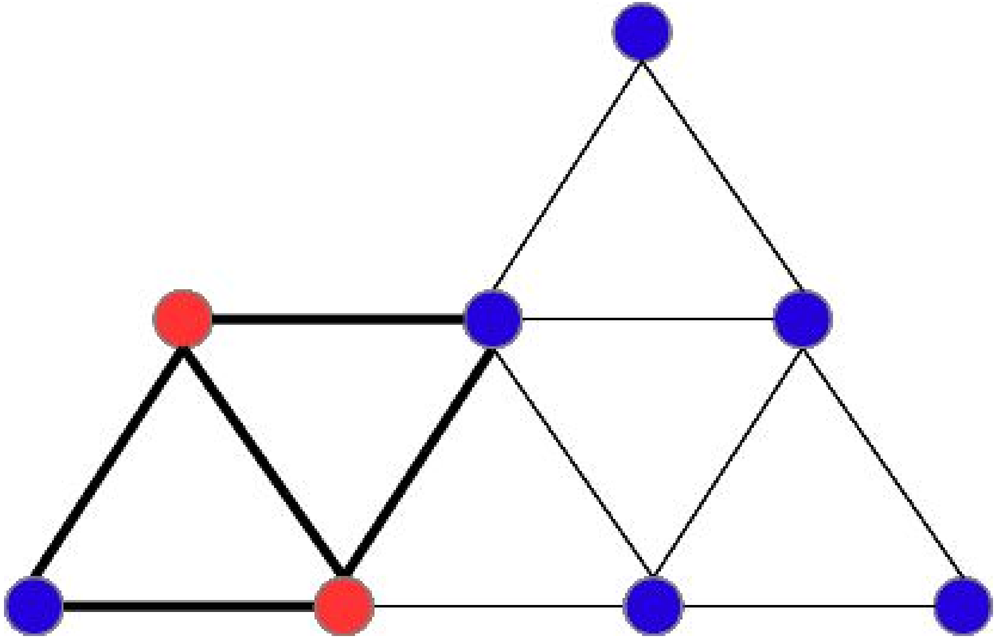
Union of k-cliques reached by adjacent k-cliques



Description of the algorithm

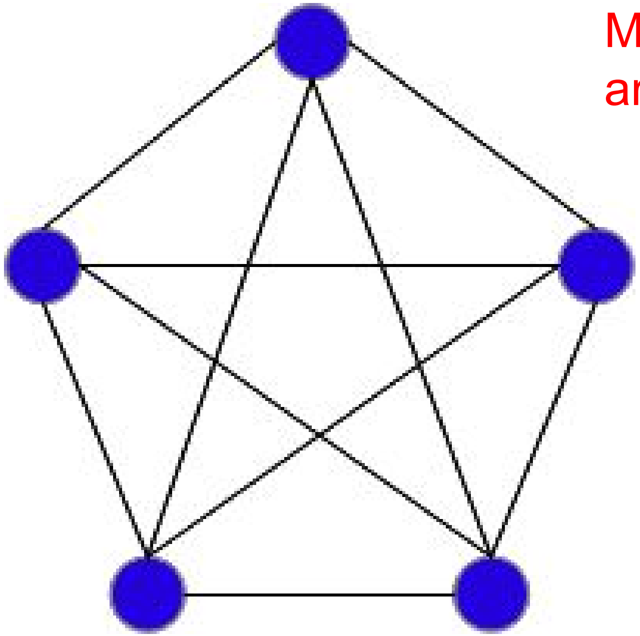
What is a k-clique community?

Union of k-cliques reached by adjacent k-cliques



Description of the algorithm

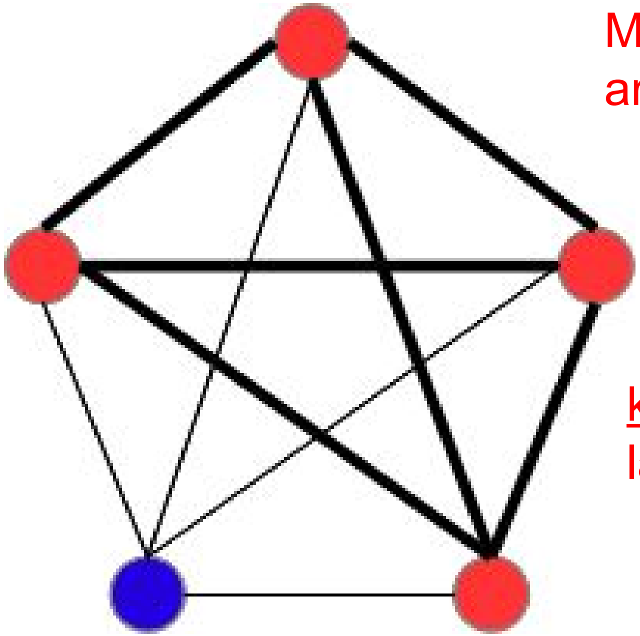
From cliques to k-cliques



Maximal complete subgraphs are called <u>cliques</u>

Description of the algorithm

From cliques to k-cliques

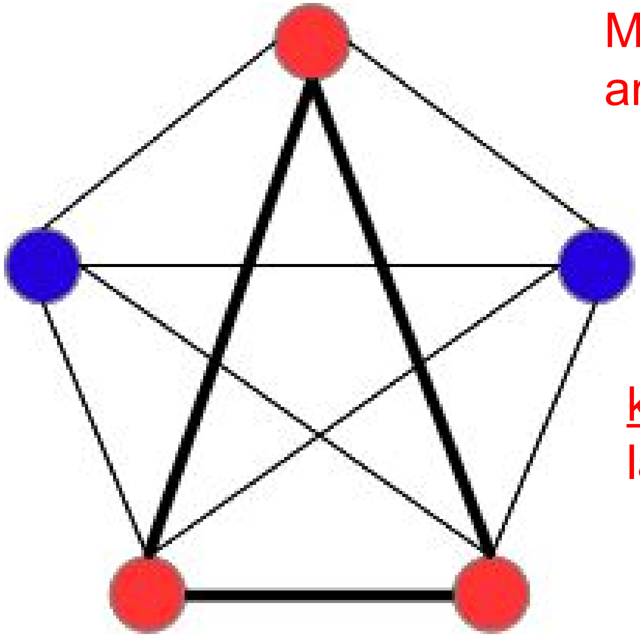


Maximal complete subgraphs are called cliques

k-cliques can be subsets of larger complete subgraphs

Description of the algorithm

From cliques to k-cliques

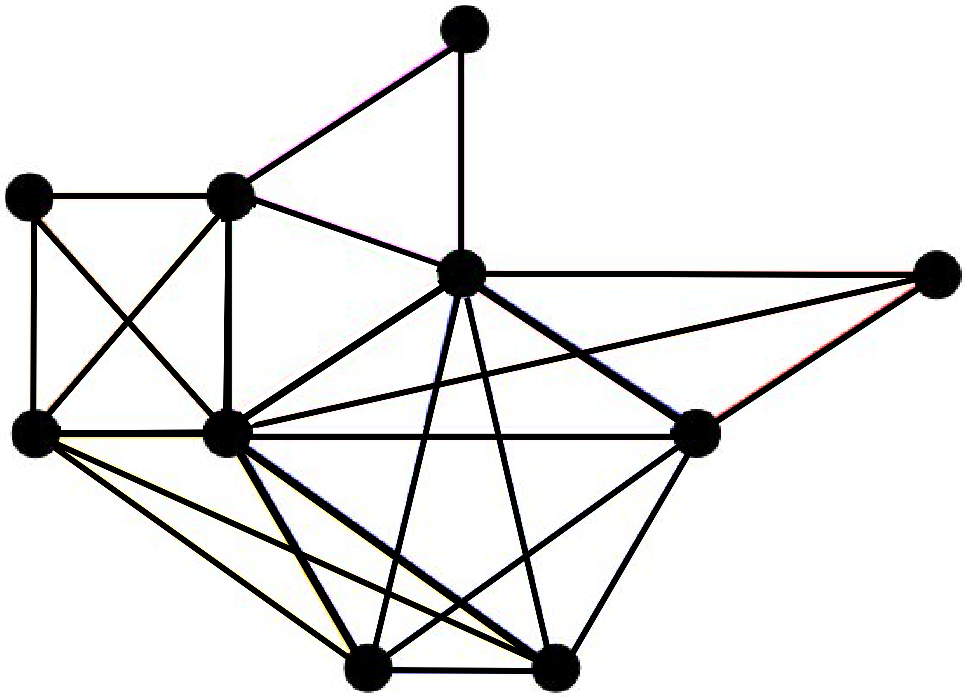


Maximal complete subgraphs are called cliques

k-cliques can be subsets of larger complete subgraphs

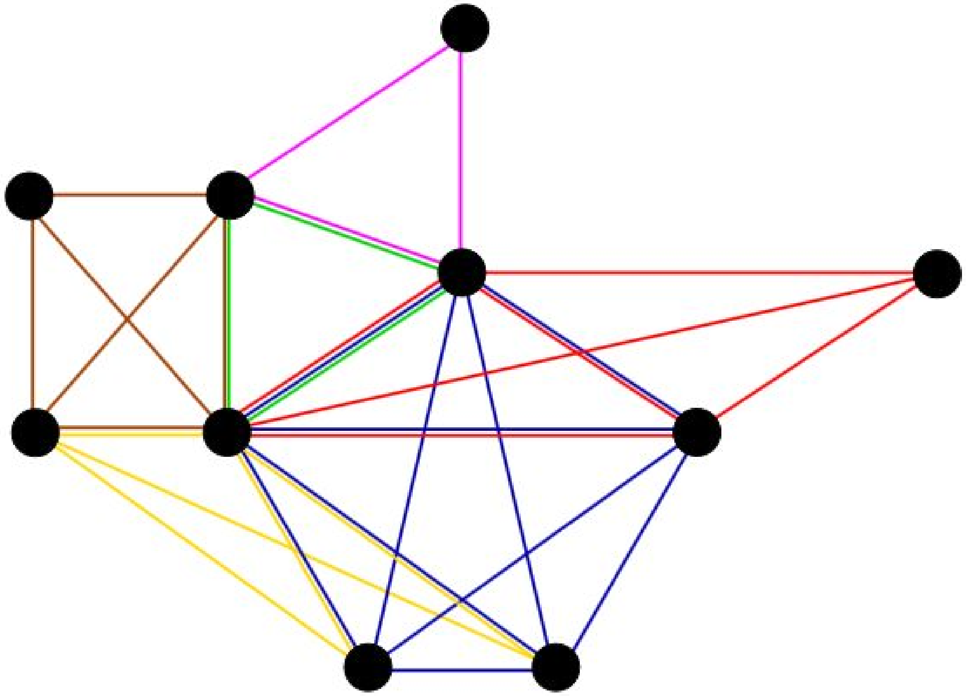
Description of the algorithm

Applying the algorithm



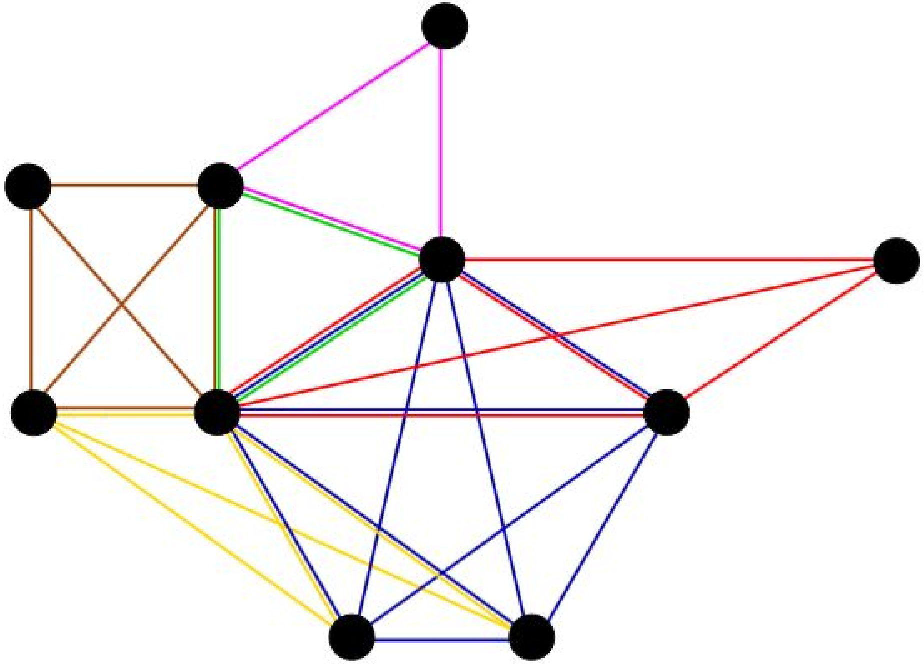
Description of the algorithm

1 - get the cliques



Description of the algorithm

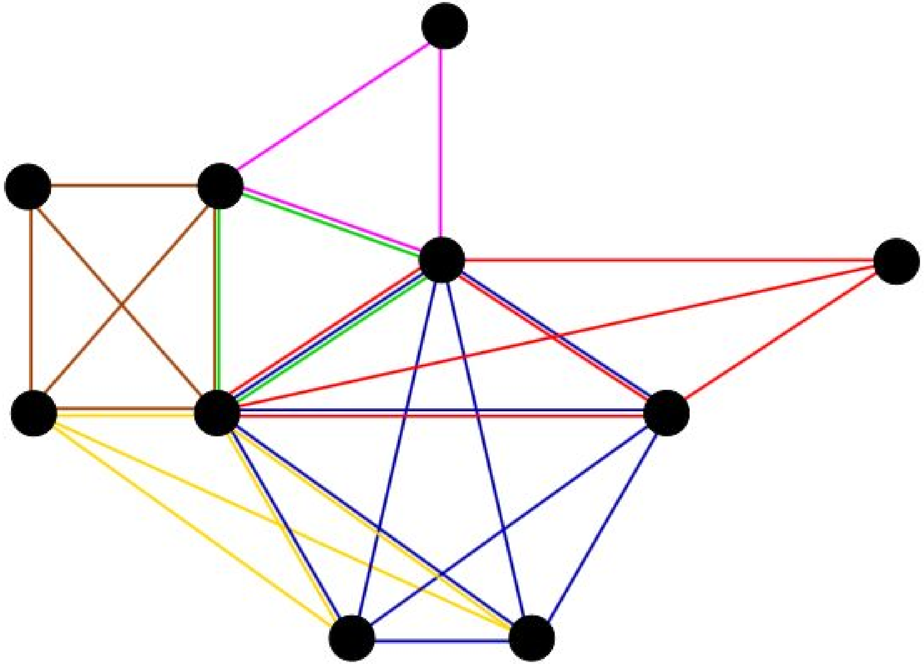
2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

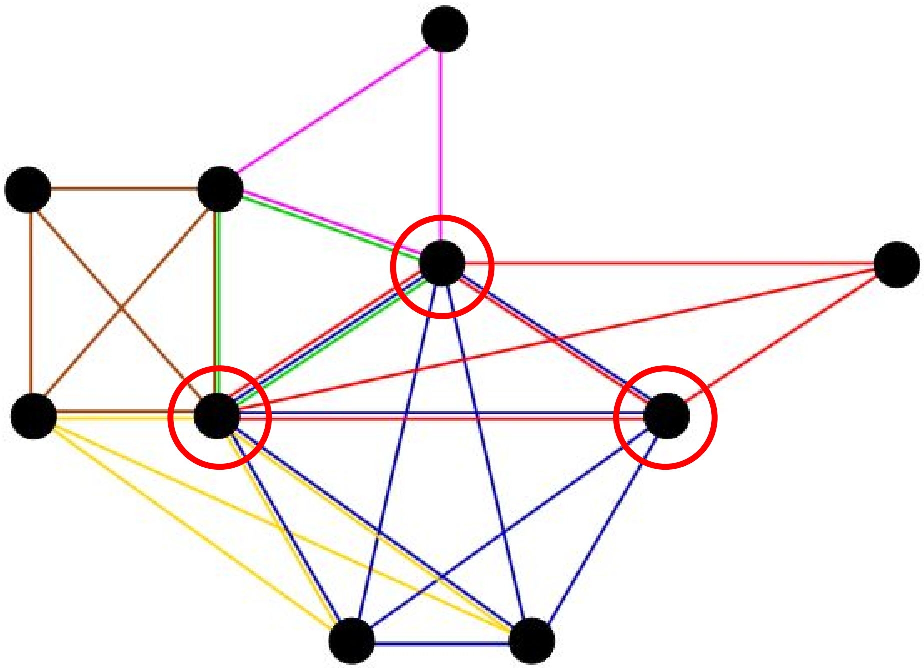
2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

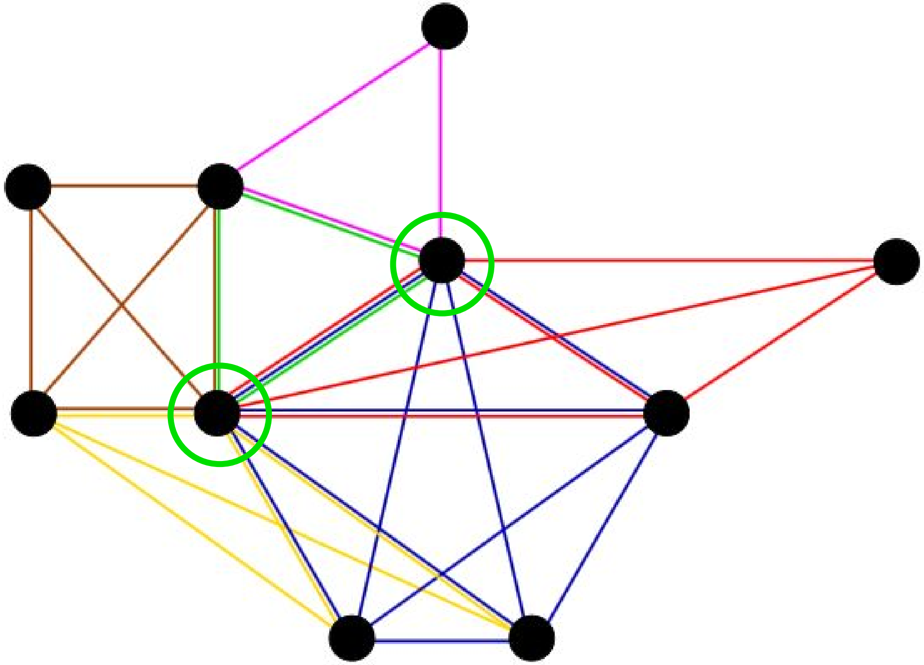
2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

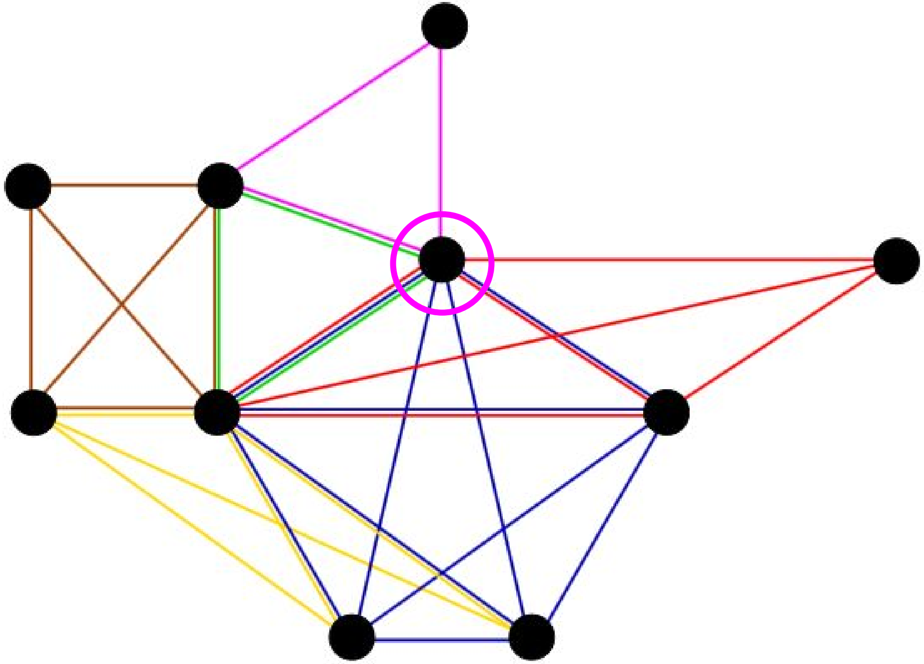
2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

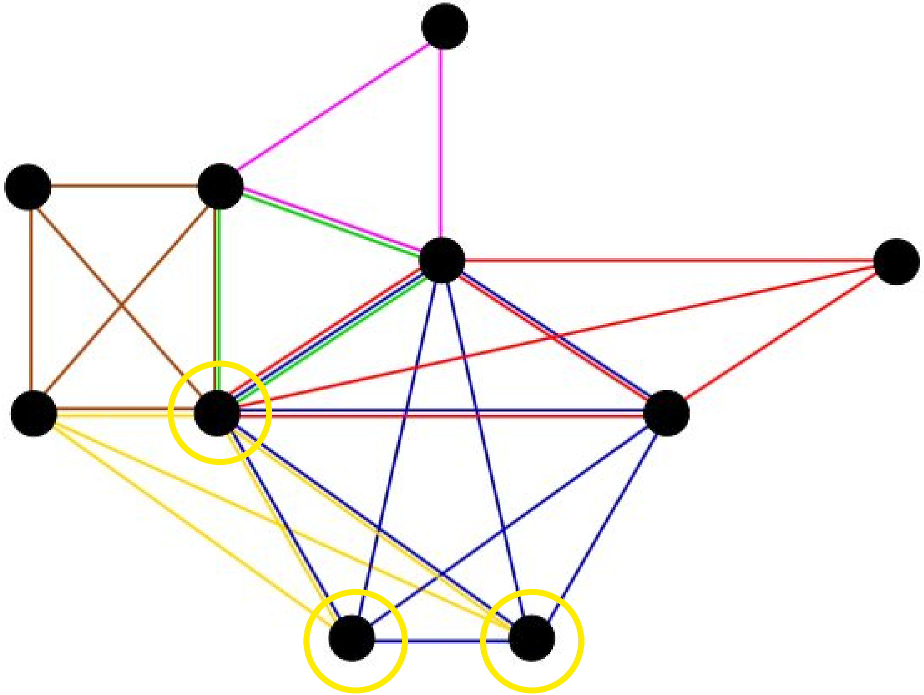
2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

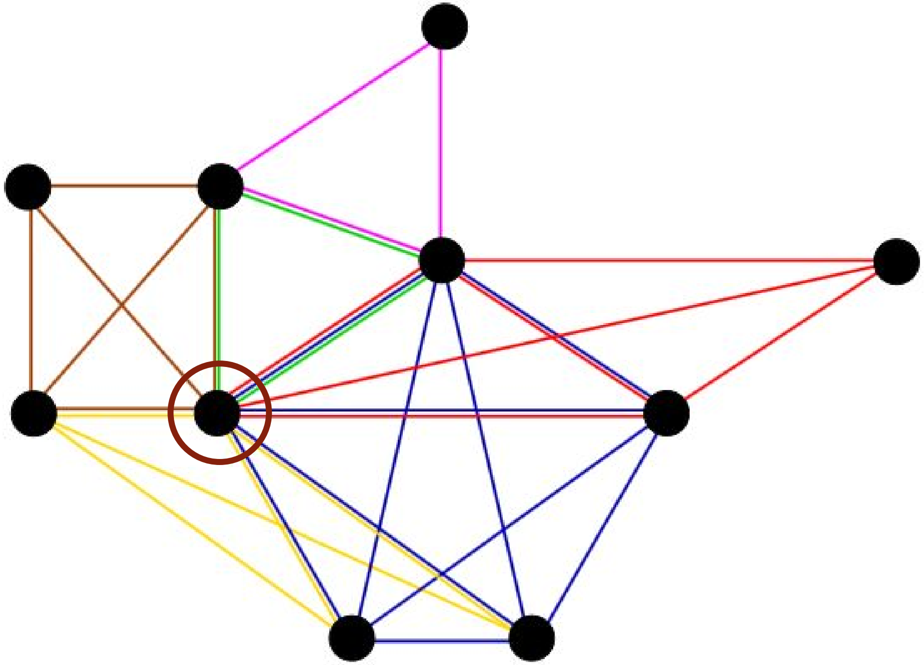
2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

2 - get the overlap matrix



	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

Description of the algorithm

3 - get the communities matrix

Delete diagonal elements that are smaller than k

Delete off-diagonal elements that are smaller than k-1

	5	3	2	1	3	1
	3	4	2	1	1	1
	2	2	3	2	1	2
	1	1	2	3	0	1
	3	1	1	0	4	2
	1	1	2	1	2	4

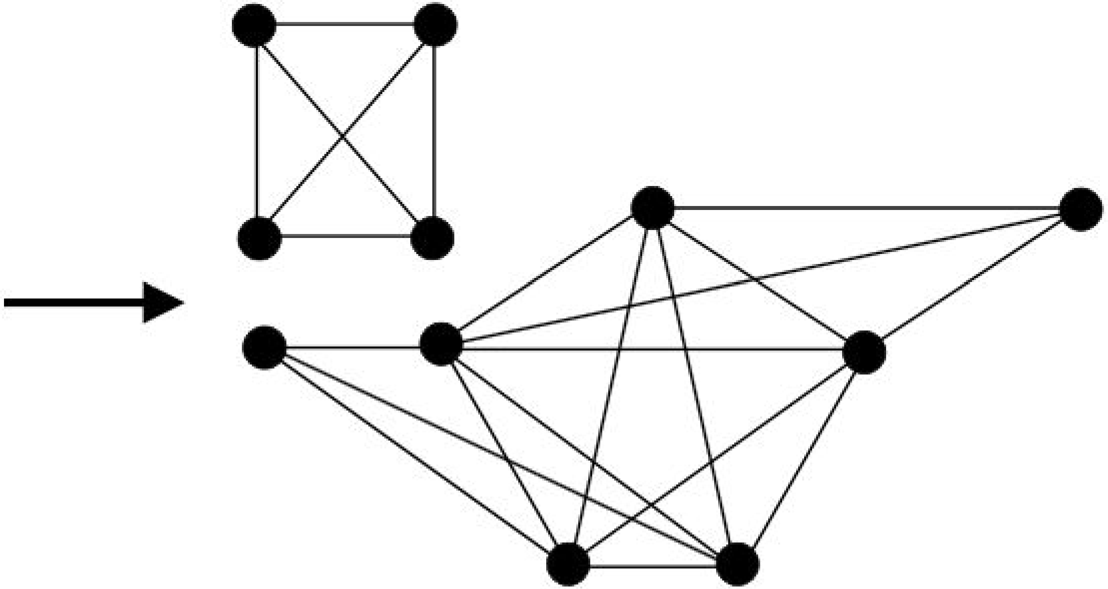
k=4 →

	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1

Description of the algorithm

4 - get the k-clique communities

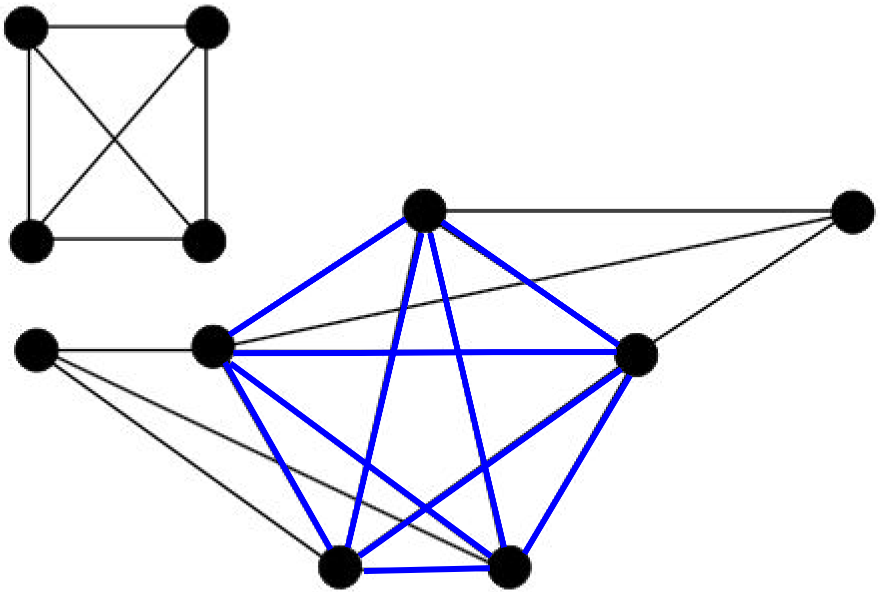
	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1



Description of the algorithm

4 - get the k-clique communities

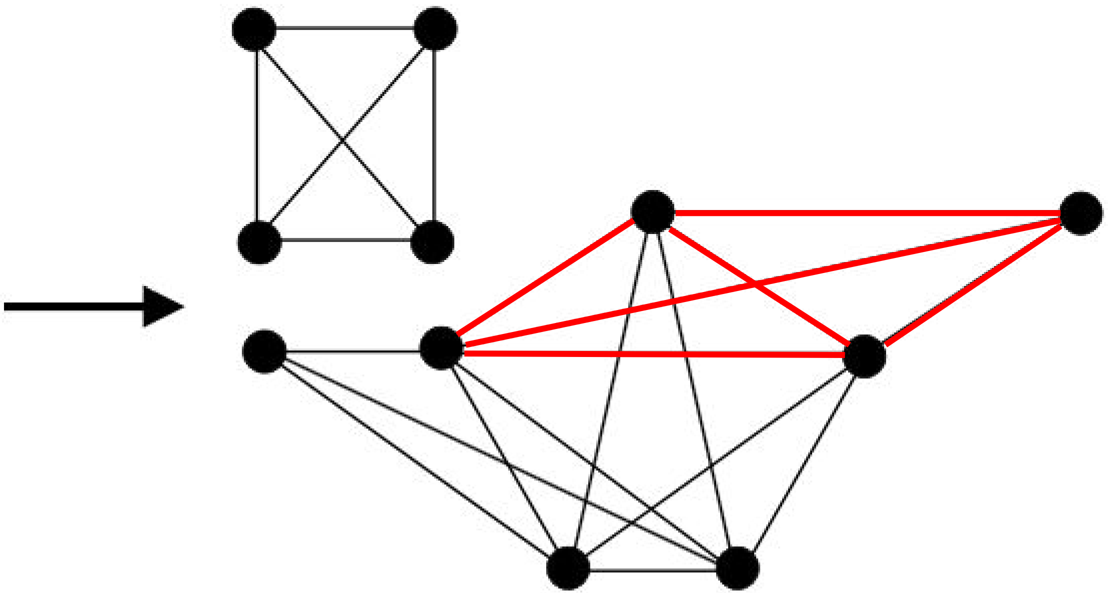
	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1



Description of the algorithm

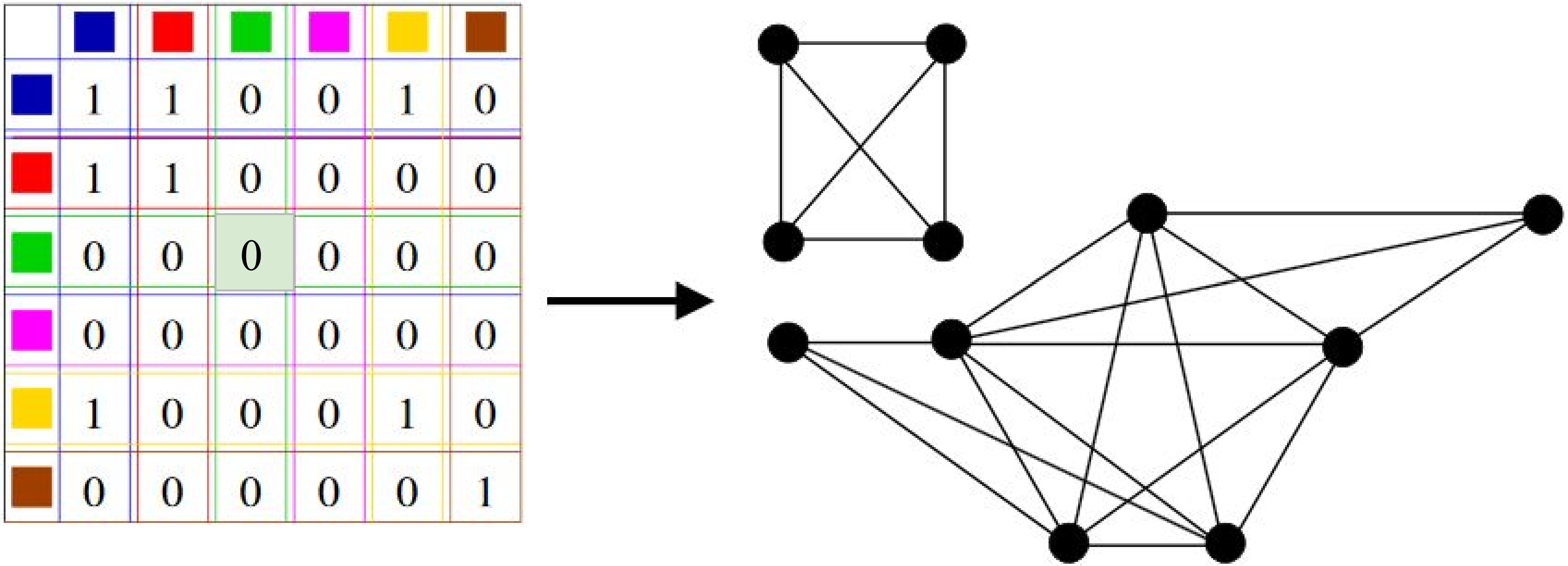
4 - get the k-clique communities

	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1



Description of the algorithm

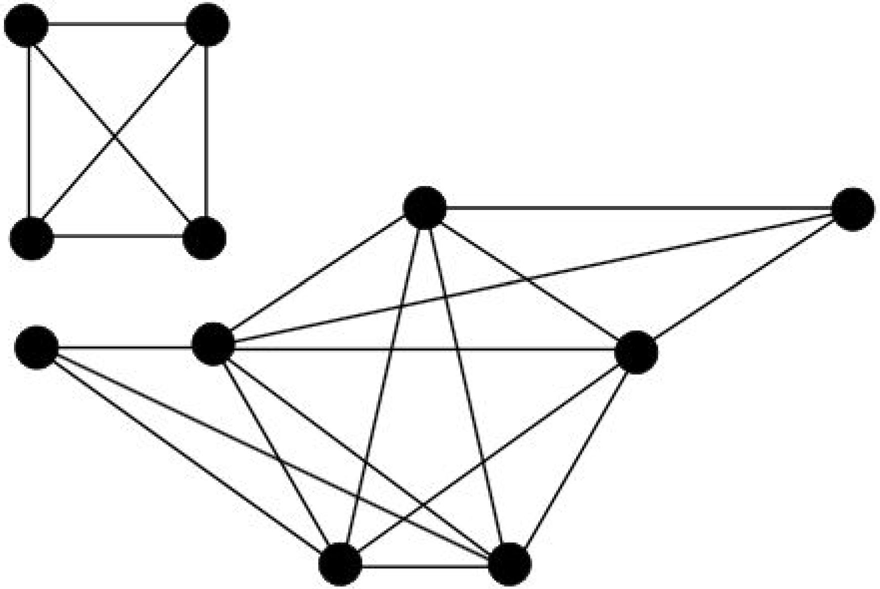
4 - get the k-clique communities



Description of the algorithm

4 - get the k-clique communities

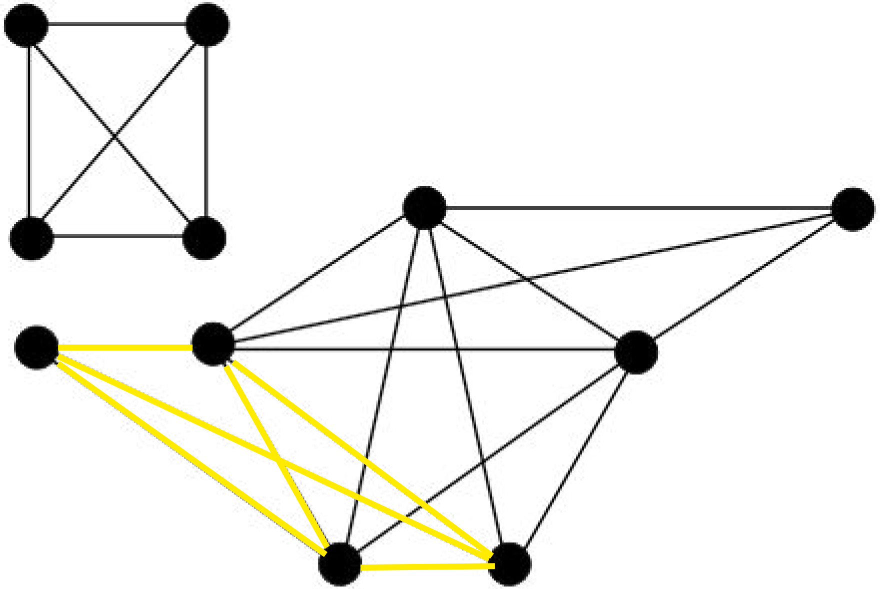
	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1



Description of the algorithm

4 - get the k-clique communities

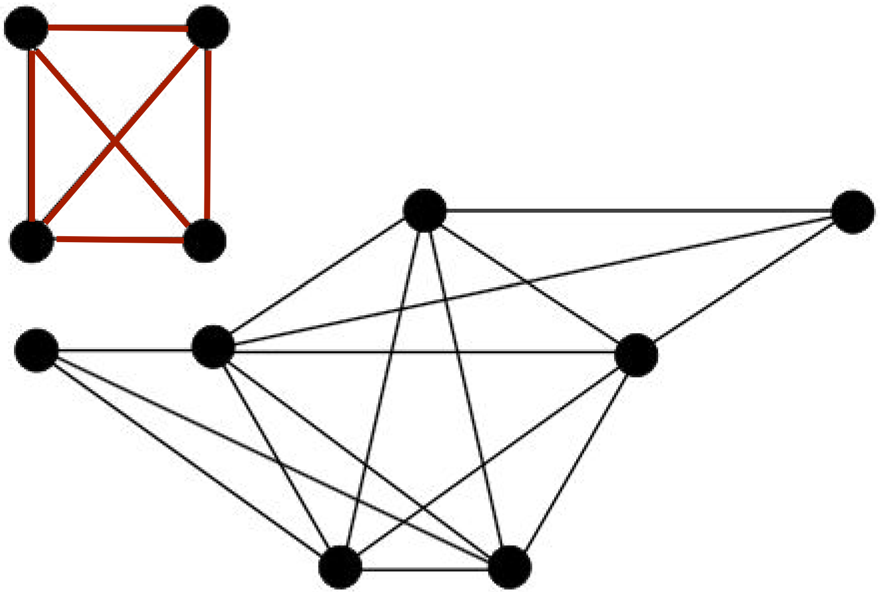
	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1



Description of the algorithm

4 - get the k-clique communities

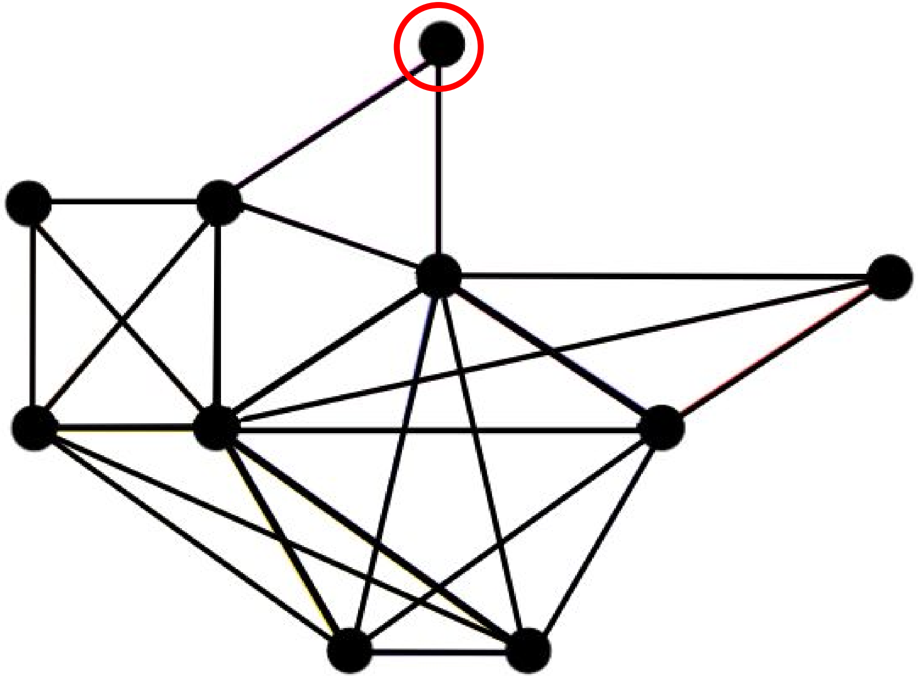
	1	1	0	0	1	0
	1	1	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	1	0	0	0	1	0
	0	0	0	0	0	1



Description of the algorithm

Important to notice:

Not all vertices will necessarily be selected to a community

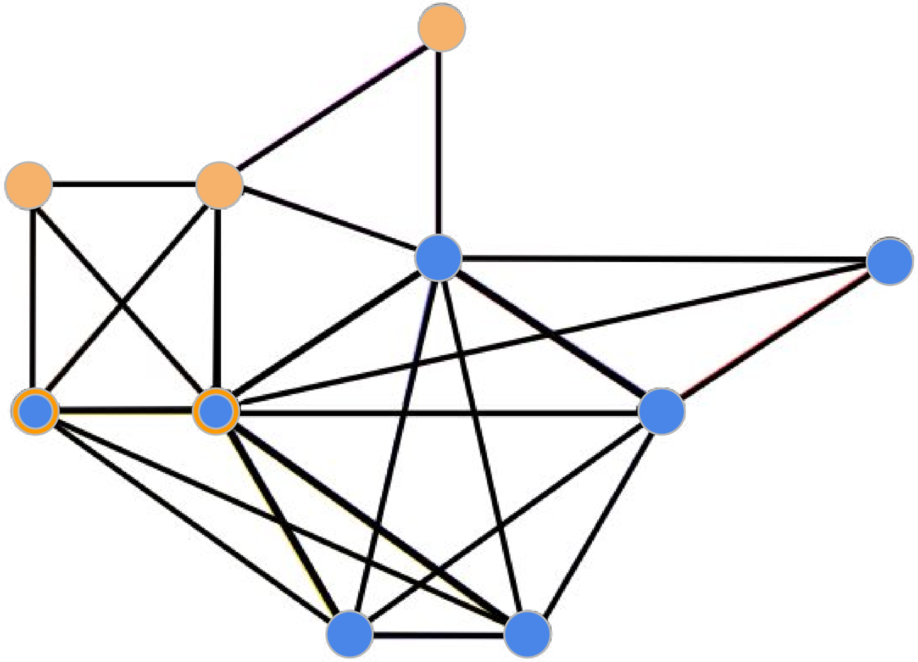


Description of the algorithm

Important to notice:

Not all vertices will necessarily be selected to a community

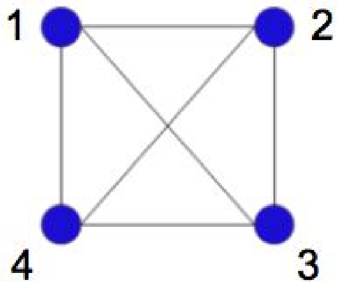
Communities might overlap

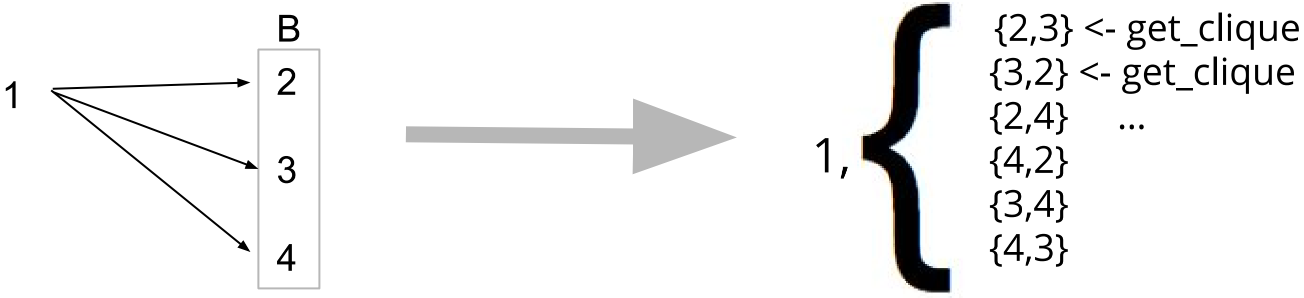


Description of the code

```
get_k_clique(k,graph):
clique = []

for node in graph:
    B = get_neighbours(node,graph)
    for subset in itertools.permutations(B, k-1):
        clique += get_clique(node,k,list(subset),graph)
```

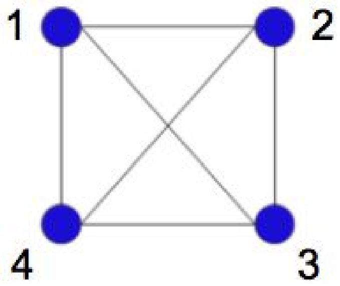




Description of the code

```
get_clique(node,k,B,graph):
A = set()
A.add(node)
while len(B) > 0 and len(A) < k:
    n = B.pop(0)
    A.add(n)
    B = list(set(B).intersection(get_neighbours(n, graph)))
    if k == len(A):
        return {" ".join(sorted(A))}

return []
```



init: A = [1]   B= [2, 3]   *node* = 1

it 1: A = [1,2]   B= [~~2~~, 3] (B = [3] ∩ [1,3,4])
it 2: A = [1,2,3] B= [~~2~~, ~~3~~] (B = [] ∩ [])

size_of(A) == K → return “1 2 3”

Complexity of the algorithm

Finding the full set of cliques on a graph:

Non-polynomial problem

Getting the overlap matrix:

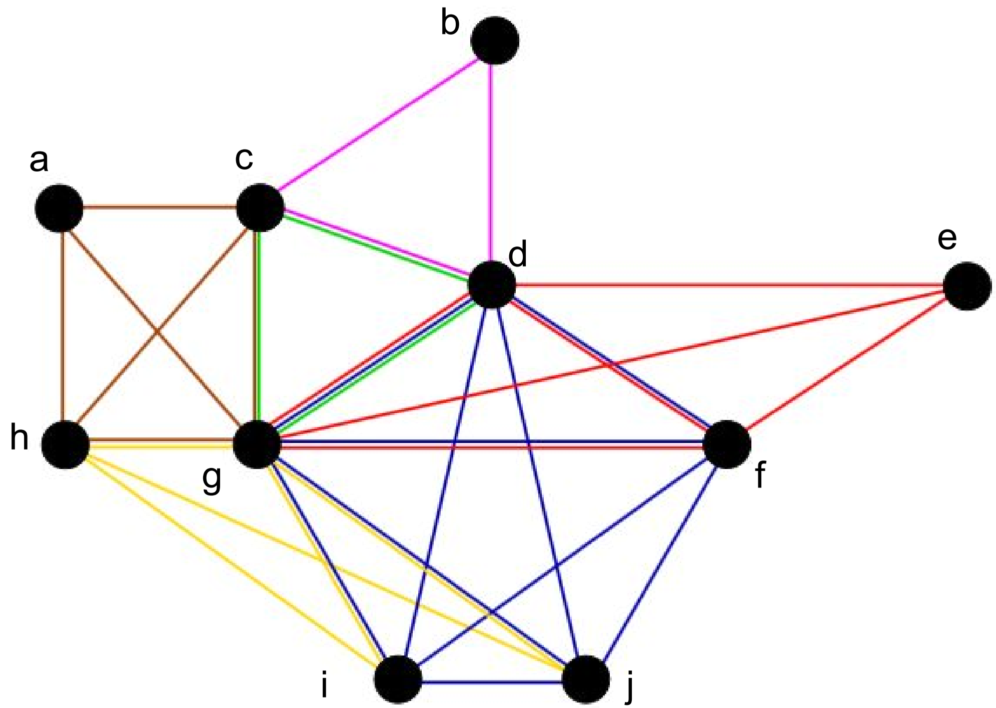
Compare each clique (c) = $O(c^2)$

Getting the k-cliques community:

Check half of the elements in the matrix = $O(c^2)$

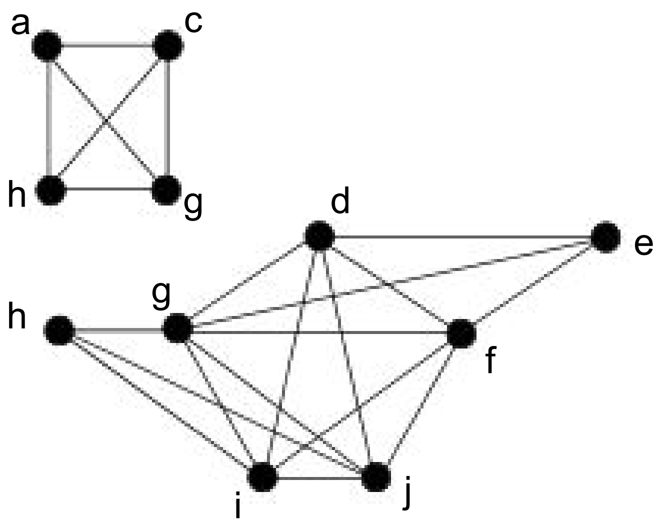
Datasets used - example 1

- First example



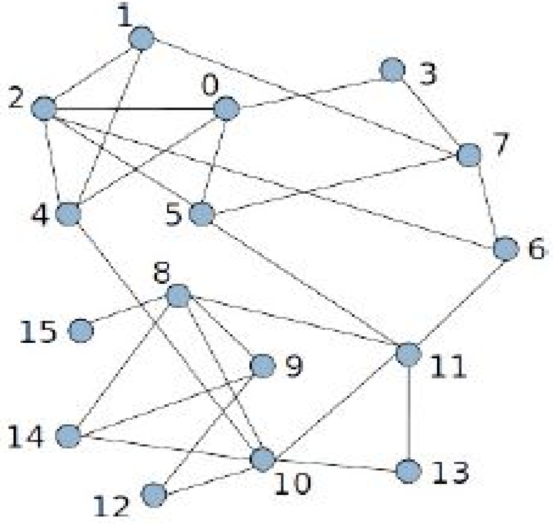
Datasets used - example 1

- Value of K : 4
- final result



Datasets used - example 2

Let’s consider the following graph :



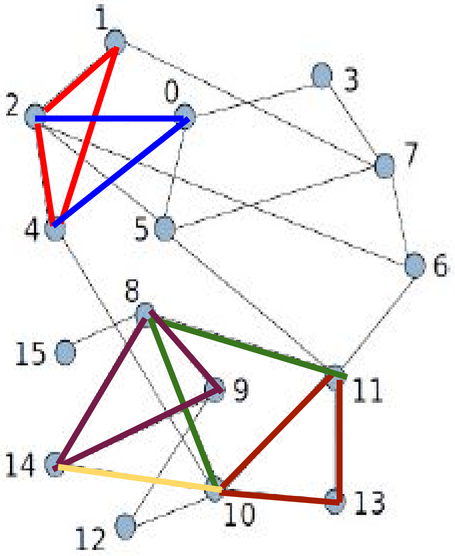
Datasets used - example 2

- k = 4 : No clique
- K = 3

1 - 2 - 4

0 - 2 - 4

8 - 9 - 14



8 - 10 - 11

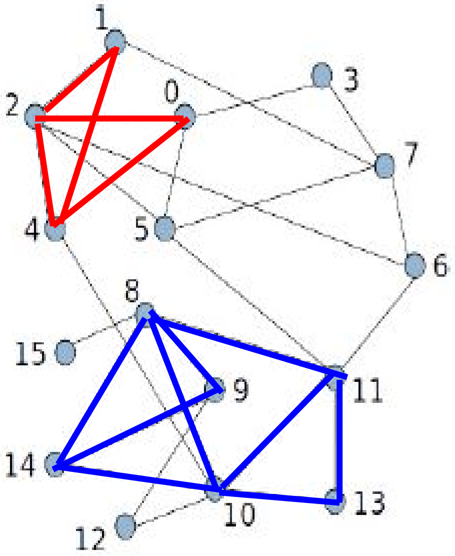
8 - 10 - 14

10 - 11 - 13

Datasets used - example 2

- final result :

2 communities



Optimization score

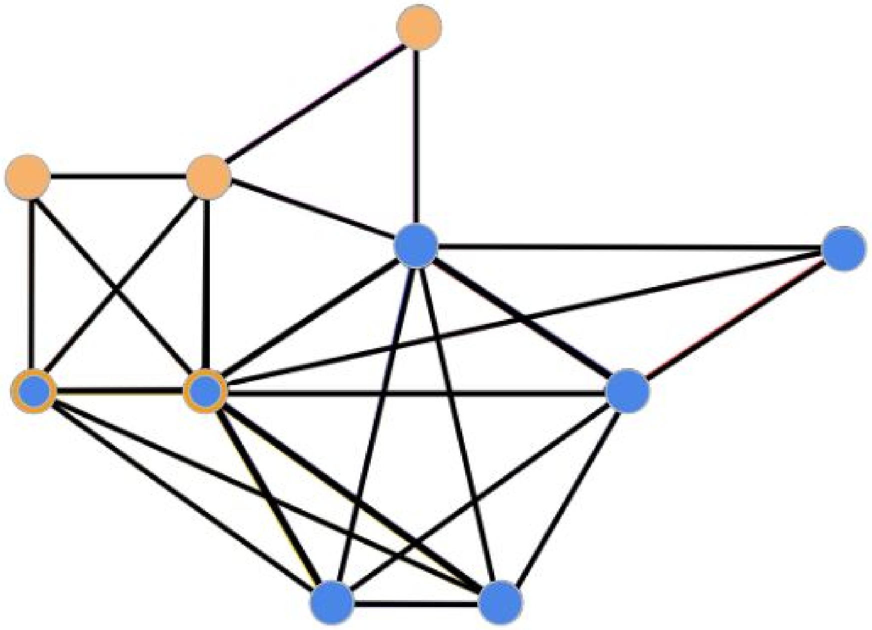
Modularity

$$Q = \frac{1}{m} \sum_{s=1}^{K} \left(m_s - \frac{d_s^2}{4m} \right)$$

m = 24 k = 2

m1 = 7 d1 = 15

m2 = 17 d2 = 36



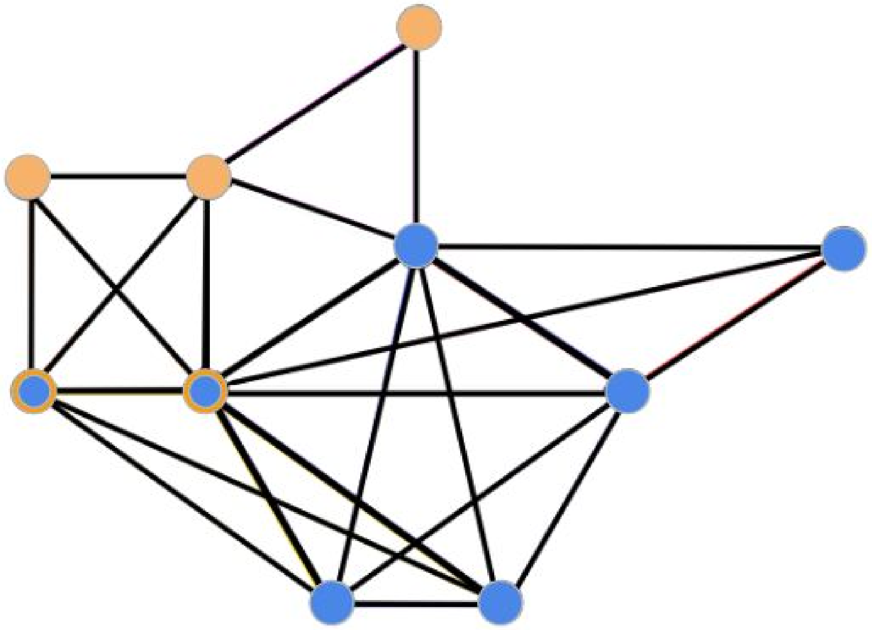
Optimization score

Modularity

Q = 0.34

The stronger the community, the higher the modularity

Influenced by the number of "single" vertices in our final result



Features of the communities

- The size of the created community is strongly (totally) related to the value of K
- There is no established method to select the best value of K
- The community size depends on which kind of community we want

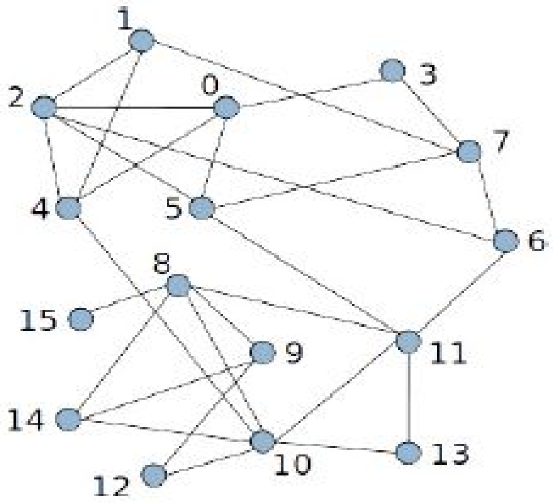
Computation times

After a computation on a very large graph, we remark that:

- the algorithm take a very long time to end
- the longest part of the algorithm is to get all the cliques

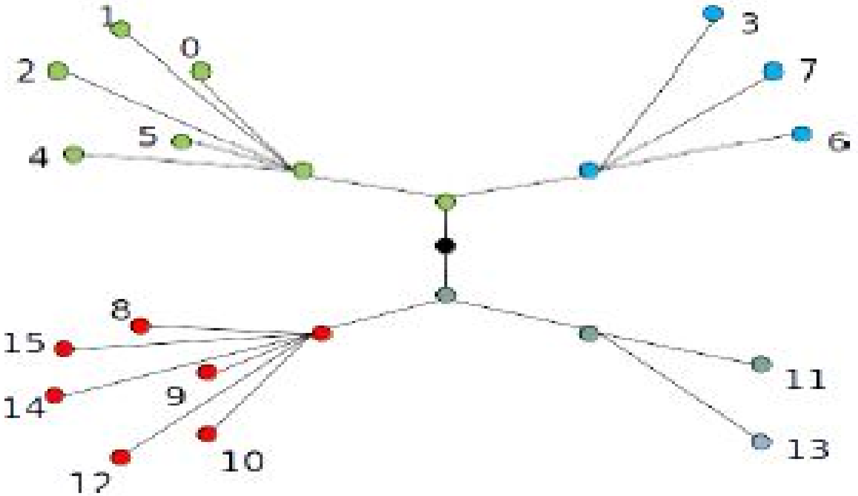
Comparison to Louvain

Let’s consider the following graph :



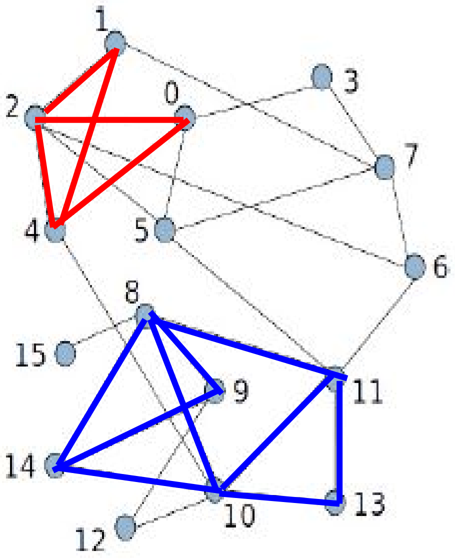
Comparison to Louvain

Community obtained with Louvain algorithm :



Comparison to Louvain

Community obtained with K-cliques algorithm :



Upgrades

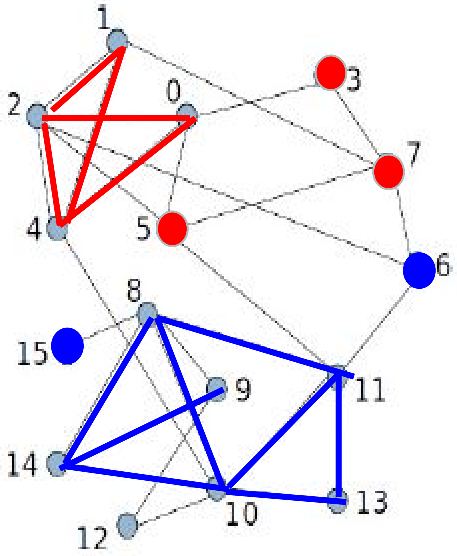
When we check for the cliques:

$$1, \left\{ \begin{array}{l} \{2,3\} \text{ <- get\_clique} \\ \{3,2\} \text{ <- get\_clique} \\ \{2,4\} \quad \dots \\ \{4,2\} \\ \{3,4\} \\ \{4,3\} \end{array} \right.$$

We could remove every similar subsets, then keep only: → {2,3} {2,4} {3,4}

Upgrades

Add the nodes which are not in any community based on the number of neighbors they have in the communities.



Questions?

UPMC

SORBONNE UNIVERSITÉS