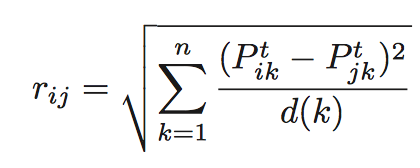
1. Algorithm

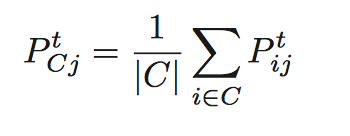
The algorithm we used in our project is based on the paper [1] with the header *“Computing communities in large networks using random walks”*[[1]](#footnote-2). The paper introduces the algorithm called *Walktrap.* Walktrap is a random walking based algorithm which is already discussed in the class by Prof. Mustafa Ozdal.

* 1. Initial Graph

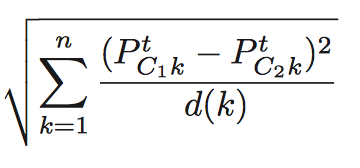
Our graph is weighted and undirected. We start with weighing the edges according to the relationship between two nodes. One node is used to indicate one person. Following relationship of a person adds the weight 0.25 and follower adds 0.25 more to the same edge if two people are in the same company or organization we assume that the relationship is stronger than a regular follower/following relationship and add 2 more to the same edge. After the initialization of the graph with the indicated parameter above *r distance* is calculated. *r distance* is another metric defined in the same paper and we are going to discuss in the next chapter.

* 1. Calculating Similarity

To find the communities, we need to be able to measure the similarities of people in the network according to the possibility of being in a specific node *k* after starting the random walk from the node *i*. As we discussed in the class, the mentioned possibility can be found by taking the *n*th. power of the weight matrix *P* for *n* being the number of steps during the random walk. After finding the probability matrix, the paper defines a new metric called *r distance.* *r distance* is calculated for each pair of nodes in the network according to their possibility of being in the node k in *n*th. step. Thus, memory requirement is O(n^2) to store *r distance*s. Mathematical definition of *r distance* can be seen in the following formula;

For the next steps of the algorithm, the same distance definition is transformed for the communities as well in a straightforward way. The probability to be in the node j in nth. step after starting from the community C is defined as follows;

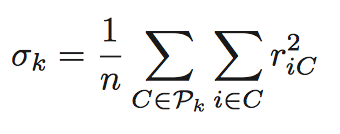
The r distance formula for communities can also be seen below;

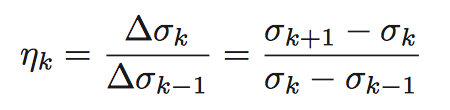


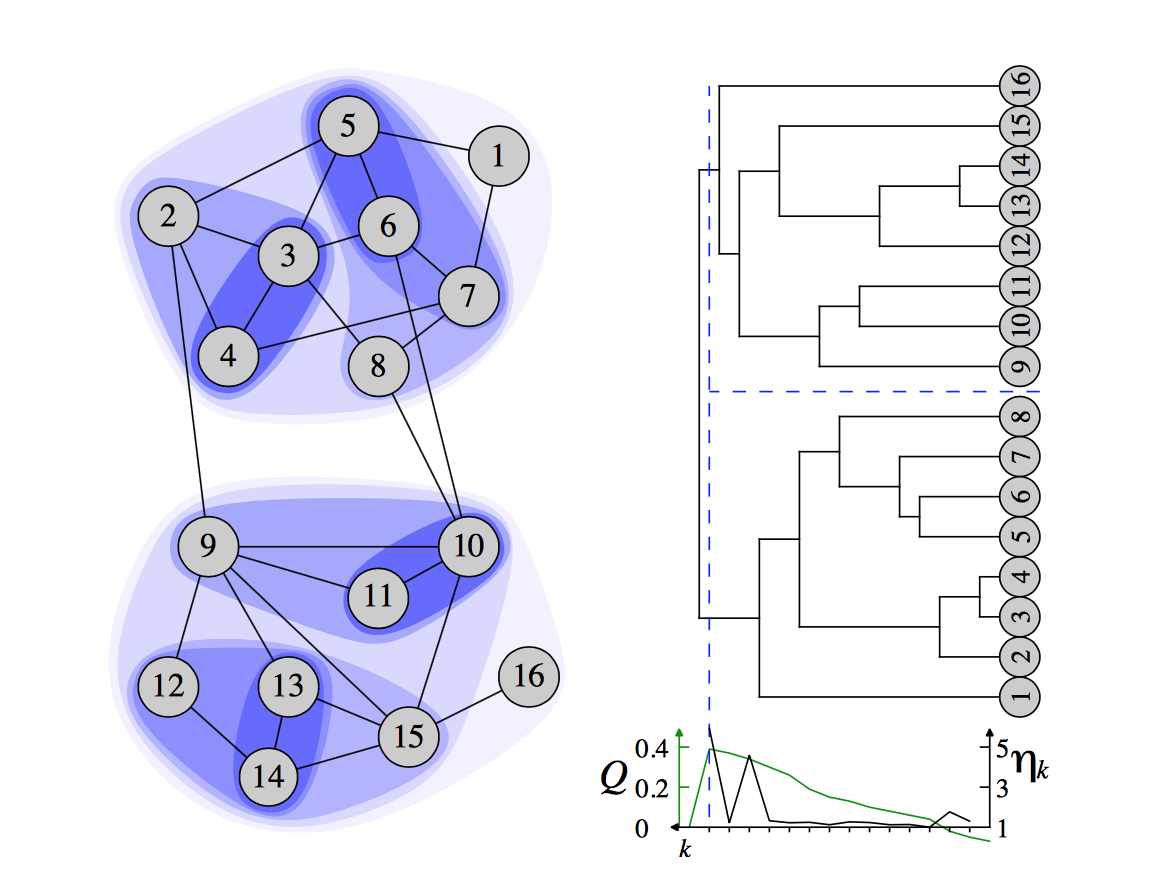
rC1C2 =

* 1. Detecting Communities

In this step, the nodes in the graph are collected together in communities according to their similarity ratios formerly calculated as *r distance.* In this step, the algorithm requires to find the adjacent communities to merge which have the closest r distance difference. The two communities are found and merged according to Ward’s method[[2]](#footnote-3). As it is stated in [1]

“At each step k, we merge the two communities that minimize the mean σk of the squared distances between each vertex and its community.”

We iteratively go through the similarity rankings and calculate σk n-1 times and each time we calculate a new metric to detect the increase ratio on σ. The metric ηk is defined as;

If ηk is large it means that the communities in step k-1 are largely relevant. After calculating ηk for each step we pick the maximal set and the output gives the communities on that step. For example, the best partition contains 2 different communities in the following example

1. <https://arxiv.org/pdf/physics/0512106v1> [↑](#footnote-ref-2)
2. <https://en.wikipedia.org/wiki/Ward's_method> [↑](#footnote-ref-3)