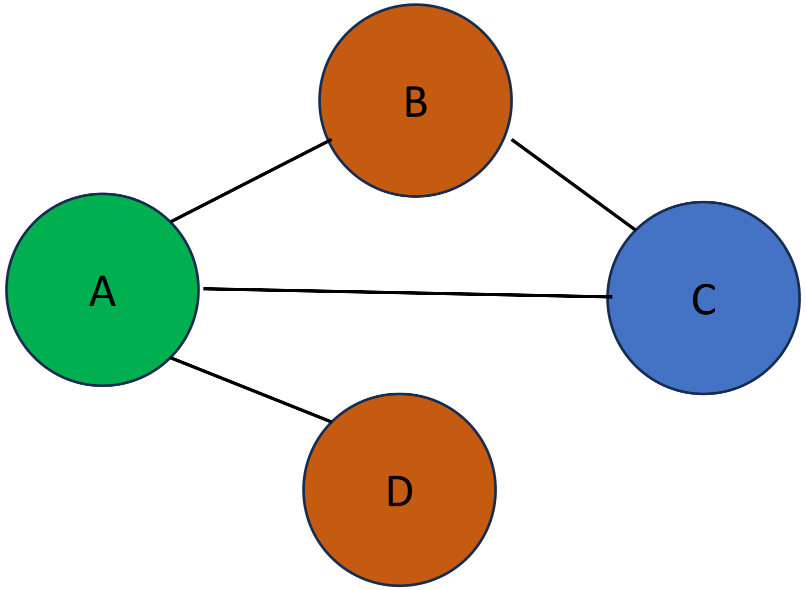
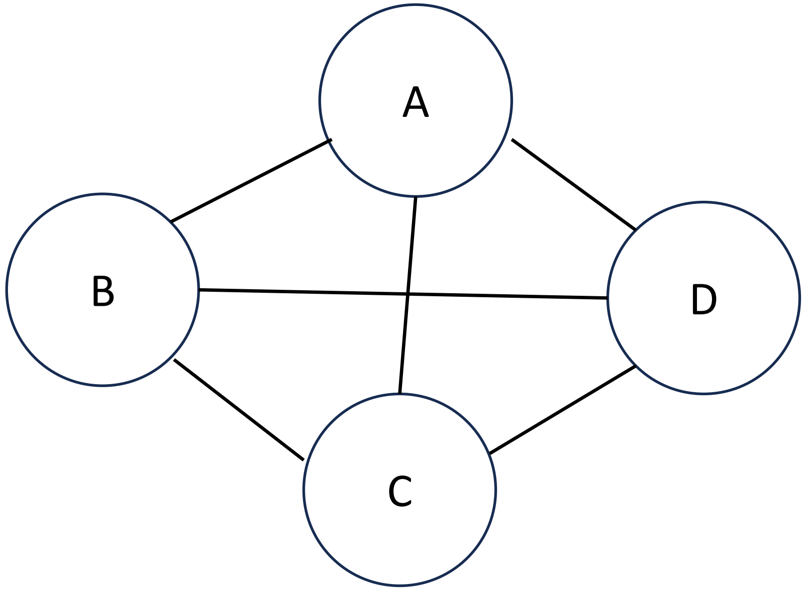
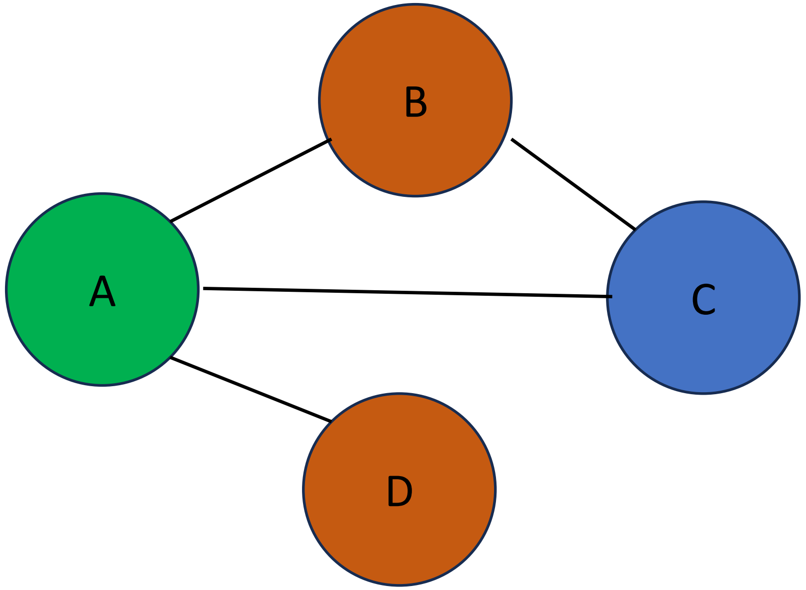
Consider the Graph Coloring problem, and assume that we want to use three colors such that connected nodes have different colors



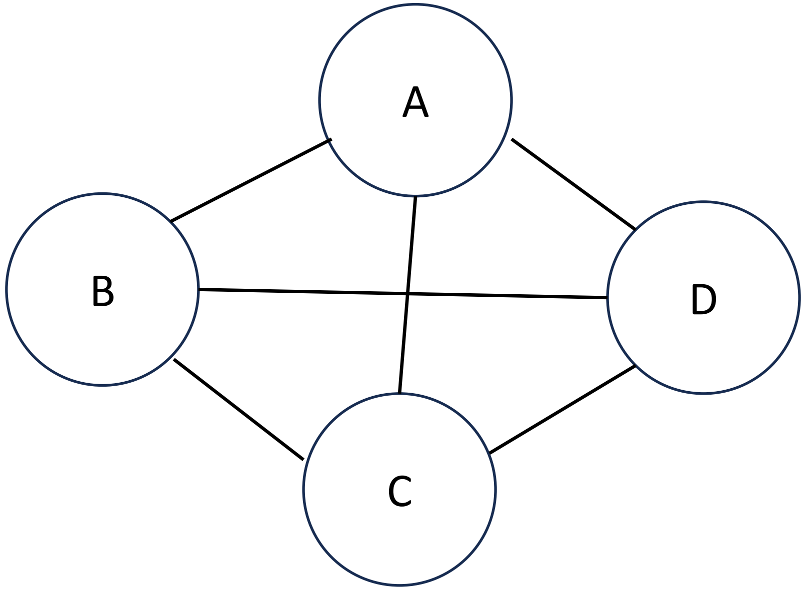
No Solution



Another way to look at the problem is to say that we want to optimize the parameters of a model of connectedness of nodes by color difference

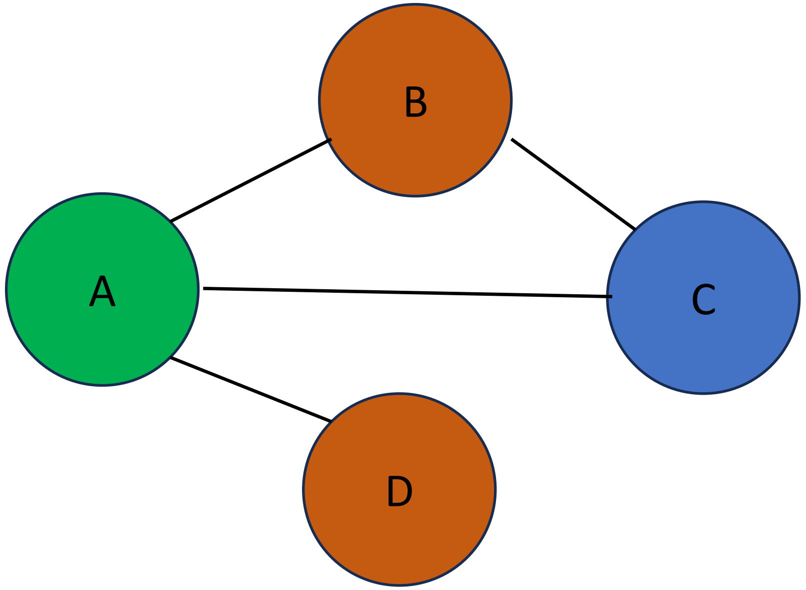


P(edge between same color)≪P(no edge between same color)



P(edge between same color)≈P(no edge between same color)

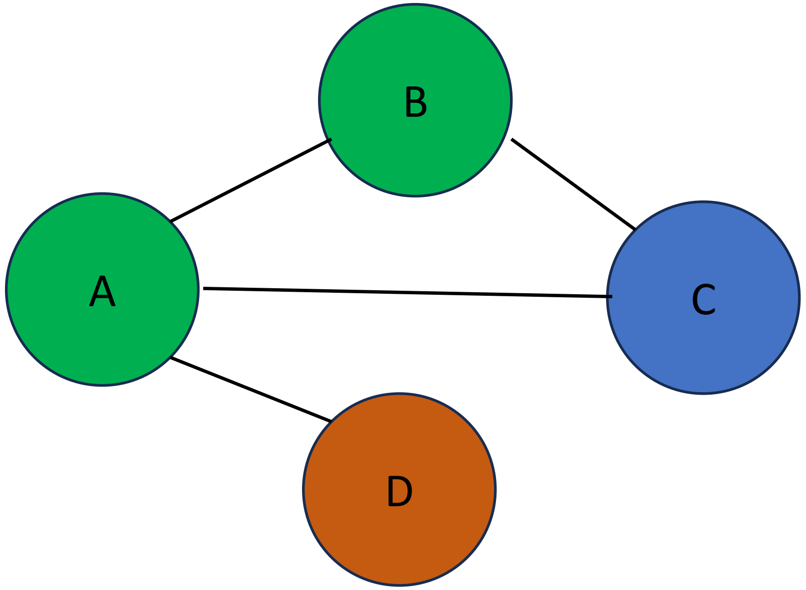
In the new formulation, we have a log-likelihood function with $\binom{n}{2}$ terms, where n is the number of nodes in the graph. We can use mini-batches to optimize it



$$log[P(A \leftrightarrow C)|A = green, C = blue]+$$
$$log[P(B \leftrightarrow C)|B = brown, C = blue]+$$
$$log\ [P(NOT\ D \leftrightarrow C)|D = brown, C = blue]$$

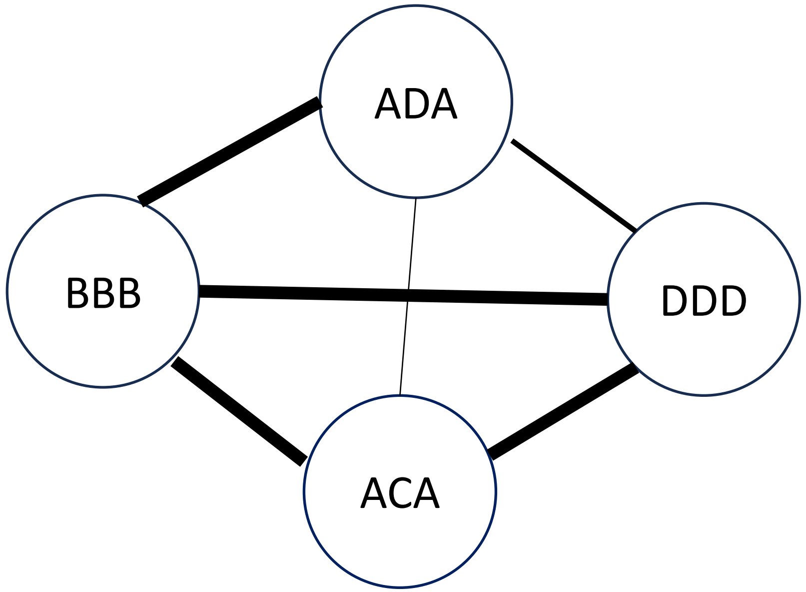
A mini-batch of 3 terms out of 6 in the log-likelihood

If node B was colored green at this iteration of the optimization, using this mini-batch will guide us to change its color to brown:

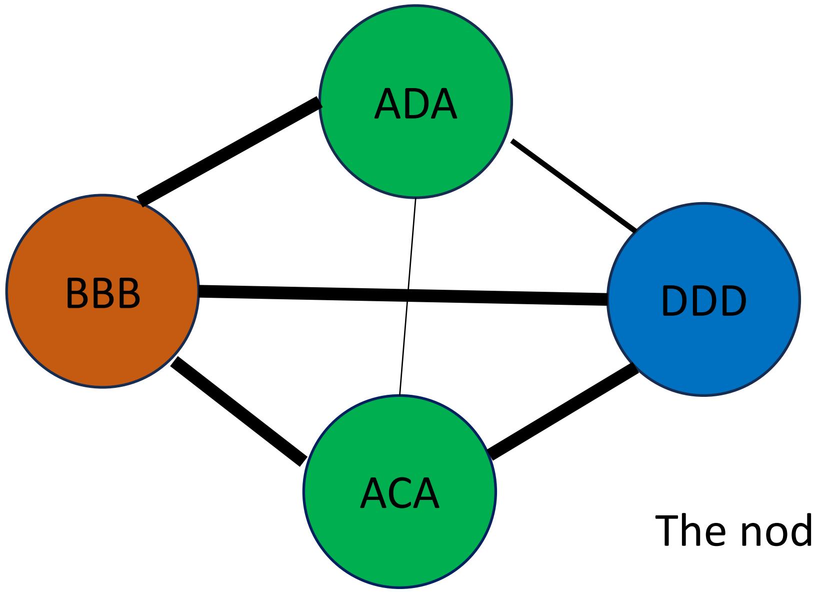


$$log[P(A \leftrightarrow C)|A = green, C = green] +$$
$$log[P(B \leftrightarrow C)|B = brown, C = blue] +$$
$$log[P(NOT\ D \leftrightarrow C)|D = brown, C = blue]$$

Consider an extension of the problem where edges have weights. For example, the nodes corresponds to strings, and the edge weights are the inverse of their similarity

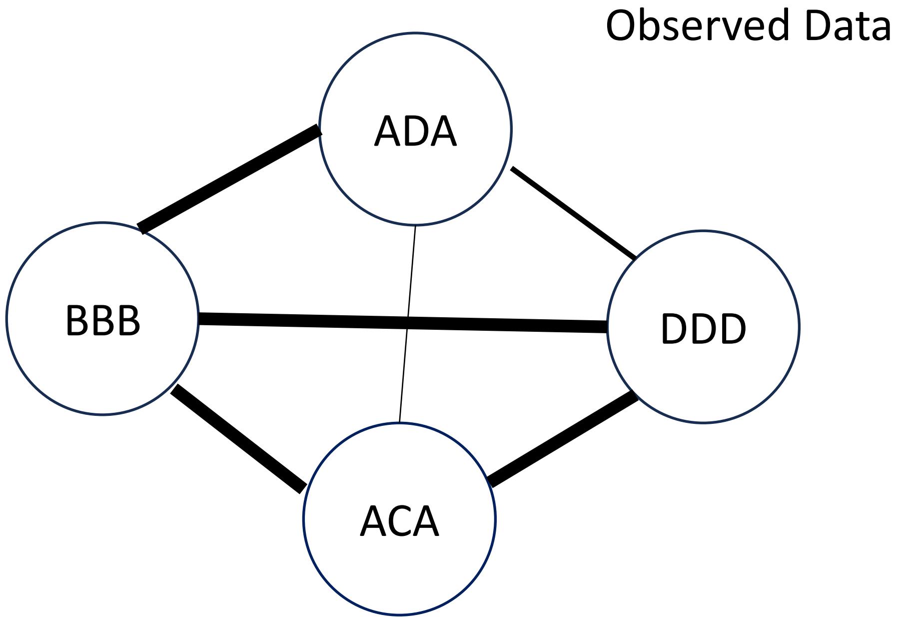


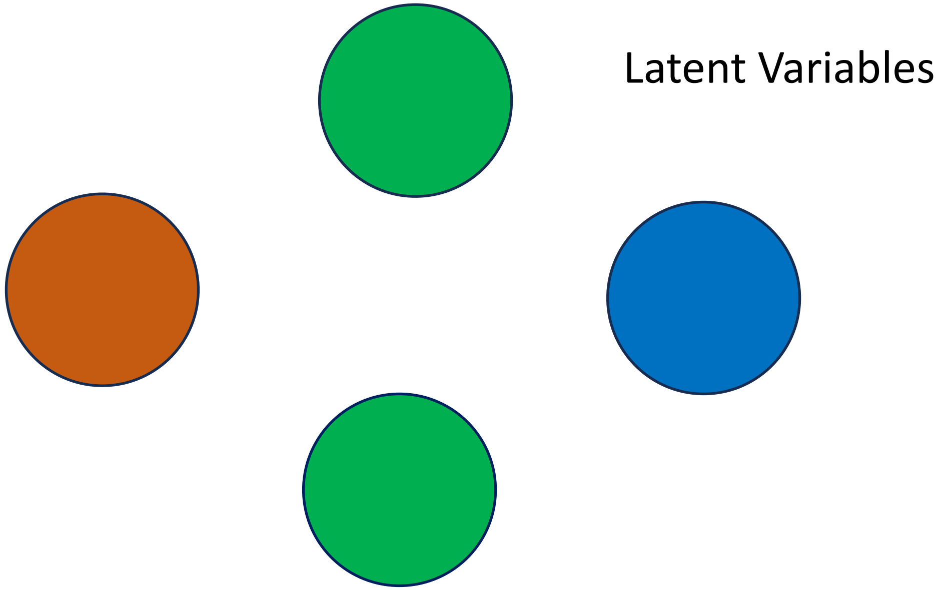
We can use the same strategy to solve the problem. The meaning of the color can be the function of the protein that the string encodes, for example



The nodes colored green have the same protein function

Two types of parameters are optimized in the new formulation of the problem, the colors (latent variables), and the model parameters, which map the coloring to probabilities given the edges





$$\sum log[P(edge)|colors]$$

model parameters