EE5239 Project - Fall 2019

Validation of Results: A Lagrangian Relaxation Network for Graph Matching [1]

MICHELLE KLECKLER and SARTHAK JAIN

INSTRUCTOR: MINGYI HONG

December 20, 2018

0.1 Introduction

A graph G is defined by a set of nodes or vertices V and a set of edges E between vertices. Graphs are a powerful tool because they can compactly and elegantly describe the structural relationships of problems engineers encounter often. The structural flexibility of representation is one of the most important advantages of using graphs. However, this flexibility is also a major difficulty since there are $O(2^n)$ subgraphs of a graph with n vertices. Graph matching (GM) is not a straightforward or easy task. [2]

Still, Graph Matching is an *important* task. GM is an essential tool used in computer vision, robotics, pattern recognition and communication networks, with vast applications in machine learning, graphics, bioinformatics, and combinatorics [3]. In simple terms, the object of the graph matching problem is to find a way to make the graphs as close as as possible by rearranging the nodes. In particular, the object is to find a bijective mapping between the nodes of the graph that minimizes the distortion (difference) or maximizes the affinity (correspondence) according to some distortion distance or correctness measure.

Graph matching (GM) problems can be divided into various categories. GM algorithms can be performed on two graphs (two-graph matching) or multiple graphs (multi-graph matching). In this paper, we focus on two graph matching.

There are two major kinds graph matching problems: (i) Exact Graph Matching (also called *isomorphism*) and (ii) Inexact Graph Matching (also called *error-tolerant* or *error-correcting GM*). [3] Exact graph matching is a special case of graph matching which occurs when the bijective mapping between the nodes in the graphs can be found with zero distortion. Graphs that can be exactly matched are called *isomorphic* and there is a strict correspondence among nodes. Inexact graph matching relaxes that constraint and requires that the distortion is minimized in the bijection when an exact isomorphism is impossible. Inexact GM has attracted much attention because of its flexibility and practicality. Many real world problems can be modeled in this way.

Finally, when each edge has a value associated with it, the problem becomes a weighted graph matching problem. In this project, we have implemented an algorithm for the two-graph matching problem for exact matching with weighted and unweighted edges.

0.2 Background

There are many ways to approach the graph matching problem and the GM problem is used for many applications.

There are special structures in which the exact matching can be found in polynomial time [3], but in general answering the question of whether an exact matching exists is in NP [4]. Subgraph isomorphism - the exact matching problem with a subset of nodes - is an NP-complete problem. Maximum common subgraph (MCS) is an NP-hard problem. The goal of MCS is to find the largest collection of nodes in the set of vertices V that are isomorphic. [3]

Many exact GM problems have applications in combinatorics and theoretical analysis, while inexact graph matching problems are easily mapped to many problems in practice. We have relaxed the constraints to account for error, noise, and distortion caused by nature and inherent to many real world applications.

We will mention two approaches to the graph matching problem and discuss where the approach presented in [1] fits.

The first approach is Affinity Maximization where we maximize the correspondence of the graphs. This is a hard problem, so heuristic methods are often applied. Other work has used the Kronecker product to maximize the affinity of two graphs. This is called Factorized Affinity in literature [5] This method uses Spectral Matching to find the largest eigenvector, utilizing the fact that the eigenvalues of a matrix do not change under simple permutation. [3]

Minimizing the Graph Edit Distance (GED) is another approach where we minimize the cost of transforming one graph into another. This transformation can occur by permutation of nodes and adding or deleting edges. Each of these operations has an associated cost [3]. This is an NP-complete problem and there are several formulation and methods such as relaxation, tree search, bipartite GED, and Quadratic Assignment Problem (QAP-)based GED to make solving feasible. [6] [7].

The first method maximizes the closeness (affinity), while the algorithm we focus on in this paper minimizes the disparities (GED). In particular this algorithm [1] uses Lagrangian relaxation to minimize GED.

0.3 Our Contribution

The contribution of this paper is validation of the algorithm proposed in [1]. We implemented the code for the algorithm and created examples to illustrate its efficacy. We also show that the algorithm is extendable to weighted graphs without modification.

0.4 Problem Formulation

We will model each graph as an adjacency matrix. The rows and columns are nodes of the graph and the entries of the matrix describe the edges. For unweighted graphs, the entries are either zero or one. The $(i, j)^{th}$ entry of the adjacency matrix is a 1 if there is an edge connecting nodes i and j, and it is 0 otherwise. For weighted graphs, the entries are the weights corresponding to each edge and 0 otherwise.

G and g are symmetric and sparse matrices. The adjacency matrices are symmetric because we are dealing with undirected graphs for this problem.

The desired permutation matrix M is a binary matrix with the property that each row and column sum to one. This is called a *doubly stochastic* matrix.

For two isomorphic graphs G and g and permutation matrix M, $GM = Mg$. Graph isomorphism is a special case of our problem, that is, we can find a permutation matrix M that makes the distance measure exactly zero. In general, we can define a distance measure between the graphs as in equation 1:

$$\begin{aligned} \text{distance} &= \|GM - Mj\|^2 \\ &= \sum_{a,i}(\sum_{b} G_{ab}M_{bi} - \sum_{j} M_{aj}g_{ji})^2 \end{aligned} \tag{1}$$

Then we seek to minimize this distance and the objective function is:

$$\begin{aligned} \underset{M}{\text{minimize}} \quad & \sum_{a,i}(\sum_{b} G_{ab}M_{bi} - \sum_{j} M_{aj}g_{ji})^2 \\ \text{subject to} \quad & \sum_{a} M_{ai} = 1, \sum_{i} M_{ai} = 1 \end{aligned} \tag{2}$$

The optimizer is the best permutation matrix M for G and g.

The above optimization problem has 2^N configurations, so the first simplification proposed in this paper is to utilize a Potts Glass approach, which is a mean field method [1] that considers the row and column constraints and reduces the number of possible configurations to N, greatly reducing the complexity from exponential to linear in N. We will use two *softmax* functions to satisfy the constraints.

In order to simultaneously satisfy the row and column constraints, the next trick is to use the Lagrangian Decomposition. Lagrangian Decomposition is achieved by separating a complicated constraint into two simpler ones and simultaneously satisfying both by equating them to eachother. For the decomposition, we create two M matrices, $M^{(1)}$ and $M^{(2)}$, and require the sum of each column of $M^{(1)}$ and the sum of each row of $M^{(2)}$ to be 1 and then set $M^{(1)} = M^{(2)}$.

The definition of *free energy* follows from the Gibbs distribution specified by an energy function, shown in equation 3.

$$F(\beta) \triangleq -\frac{1}{\beta}\log[Z(\beta)] = \mathbb{E}[E(S)] - \frac{1}{\beta}W(\beta) \tag{3}$$

In equation 3, β is the inverse temperature, $Z(\beta)$ is the partition function dependent on the energy function $E(S)$ and β. $W(\beta)$ is the entropy. Minimizing F as β approaches infinity approaches the minimum energy $E(S)$. This minimizes the distance between the graphs and solves the problem. This is called deterministic annealing.

After simplification, a Legendre transformation, and the introduction of a new parameter μ to cause the energy function to be linear with respect to M and weakly convex with respect to μ and λ, we finally arrive at the final form of the free energy function shown below in equation 4

$$\begin{aligned} F(\mu,\lambda,\sigma) = \frac{1}{2}\sum_{a,i}\mu_{ai}^2 - \frac{\gamma}{2}\sum_{a,i}\sigma_{ai}^2 + \frac{1}{\beta}\sum_{i}\log\sum_{a}\exp\left(-\beta[\sum_{b}G_{ab}\mu_{bi} + \lambda_{ai} - \frac{\gamma}{2}\sigma_{ai}]\right) \\ + \frac{1}{\beta}\sum_{a}\log\sum_{i}\exp\left(\beta[\sum_{j}g_{ij}\mu_{aj} + \lambda_{ai} + \frac{\gamma}{2}\sigma_{ai}]\right) \end{aligned} \tag{4}$$

The four components of the energy function are (1) the distance measure that we are minimizing (2) the Lagrangian Decomposition variables $M^{(1)}$ and $M^{(2)}$ (3) the equality constraint $M^{(1)} = M^{(2)}$ using the Lagrangian multiplier λ and (4) a self amplification term to break symmetries and avoid local minima of the the objective when multiple isomorphisms may exist. The algorithm involves minimizing with respect to μ and λ and maximizing with respect to σ.

To summarize, we started with a probabilistic paradigm (Gibbs free energy distribution), made some simplifications and introduced a new set of variables to make the problem convex, and ultimately descended on that convex objective function using the method called deterministic annealing. Increasing β decreases the "temperature" and the energy converges to the minimum and the best matching is achieved by the output of the algorithm M.

0.5 Implementation

The algorithm in [1] consists of the following steps. (Please note that this algorithm has been implemented in the FinalProject5239.m file)

(i) **Random Graphs:** First we create random unweighted and weighted isomorphic graphs with parameters N and c (where N is the number of vertices and c is the approximate percent of edges.

(ii) **Initialization:** We then initialize μ and λ with vectors of small positive values and σ with a zero vector.

(iii) **Descent step:** We then minimize the free energy function (equation 4) with respect to μ and λ, keeping σ constant. Note the the free energy function is convex in μ and λ and therefore we use fmincon function of matlab for minimization

(iv) **Ascent Step:** Next we update σ according to the following equation:

$$\sigma = \frac{M^{(1)} + M^{(2)}}{2} + r;$$

where $M^{(1)}$ and $M^{(2)}$ are the two mirror matrices given by:

$$M^{(1)} = \frac{\exp\left(-\beta[\sum_c G_{ac}\mu_{ci} + \lambda_{ai} - \frac{\gamma}{2}\sigma_{ai}]\right)}{\sum_b \exp\left(-\beta[\sum_c G_{ac}\mu_{ci} + \lambda_{ai} - \frac{\gamma}{2}\sigma_{ai}]\right)}$$

$$M^{(2)} = \frac{\exp\left(-\beta[\sum_k g_{ik}\mu_{ak} + \lambda_{ai} + \frac{\gamma}{2}\sigma_{ai}]\right)}{\sum_j \exp\left(-\beta[\sum_k g_{jk}\mu_{ak} + \lambda_{aj} + \frac{\gamma}{2}\sigma_{ai}]\right)}$$

and r is a random number between $[-\epsilon, \epsilon]$

(v) β **Update:** We increase β according to the following equation:

$$\beta^{k+1} = (1-\delta)\beta^{(k)} + \delta\beta_{max}$$

For this work, we have taken $\beta_{max} = 100$, $\beta^0 = 1$ and $\delta = 0.01$.

(vi) **Stopping Criteria:** If (1) $||GM^{(1)} - M^{(2)}g|| < \epsilon$ or (2) $\beta < \beta_{max}$, we stop. Otherwise we go back to step (iii). If the algorithm terminates due to condition (1), the obtained solution is optimal, if the algorithm terminates due to condition (2), the solution may or may not be optimal.

0.6 Results

First, we give a few examples of both isomorphic graphs and weighted graph matching to illustrate how the algorithm performs in each case.

In figure 1, we present two randomly generated isomorphic graphs with 5 vertices and 5 edges. For these graphs, the permutation matrix M=M1=M2 generated by the algorithm is as follows:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This permutation matrix implies that the 1st vertex in graph 'G' corresponds to 1st vertex of graph 'g', the 2nd vertex of 'G' corresponds to the 3rd vertex of 'g', etc. Hence, the correspondence between the two graphs can be summarized as $1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 2, 5 \rightarrow 5$, which is the correct isomorphism between the two graphs.

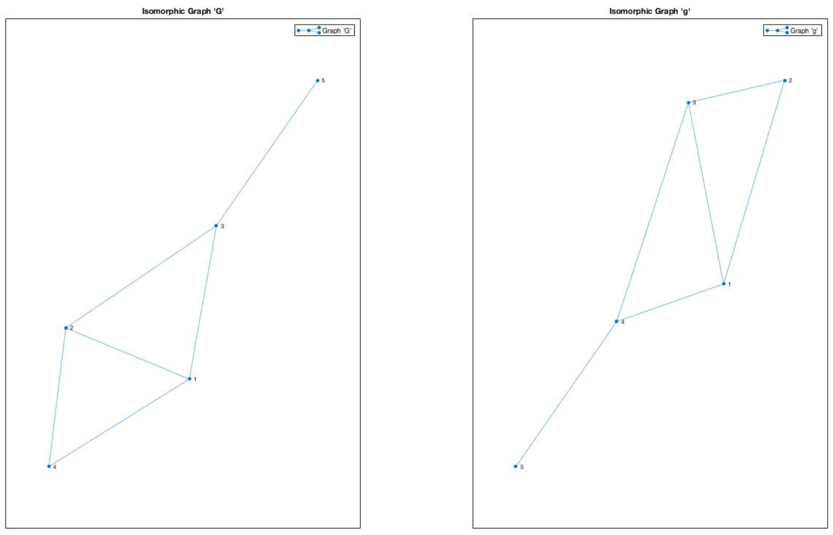

Figure 1: Isomorphic graphs consisting of 5 nodes

We do a similar analysis for a randomly generated 10 nodes isomorphic pair of graphs, presented in Figure 2. The permutation matrix for this isomorphism is given by:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The correspondences between the two graphs is given by $1 \to 8, 2 \to 2, 3 \to 10, 4 \to 6, 5 \to 4, 6 \to 9, 7 \to 7, 8 \to 5, 9 \to 1, 10 \to 3$:

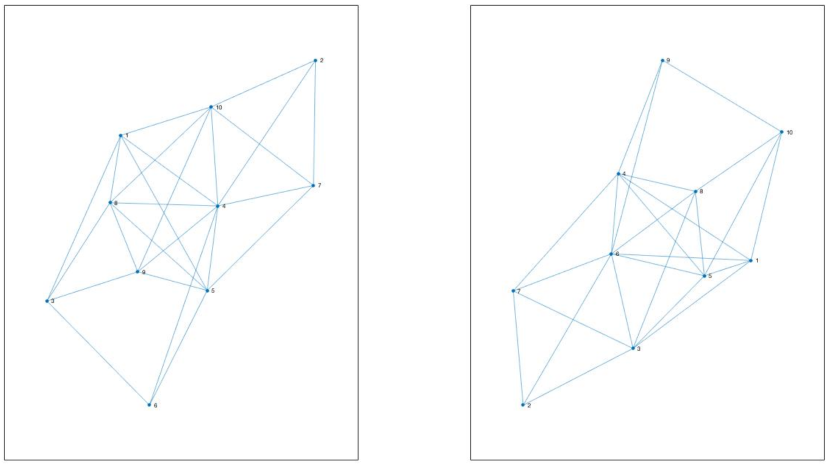

Figure 2: Isomorphic graphs consisting of 10 nodes

Next we illustrate the performance of our algorithm on a randomly generated weighted graph. Figure 3 consists of two randomly generated weighted graphs of 10 nodes each. The permutation matrix given by the algorithm is:

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The correspondences between the two graphs are: $1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 5, 4 \rightarrow 10, 5 \rightarrow 1, 6 \rightarrow 8, 7 \rightarrow 2, 8 \rightarrow 9, 9 \rightarrow 7, 10 \rightarrow 6$.

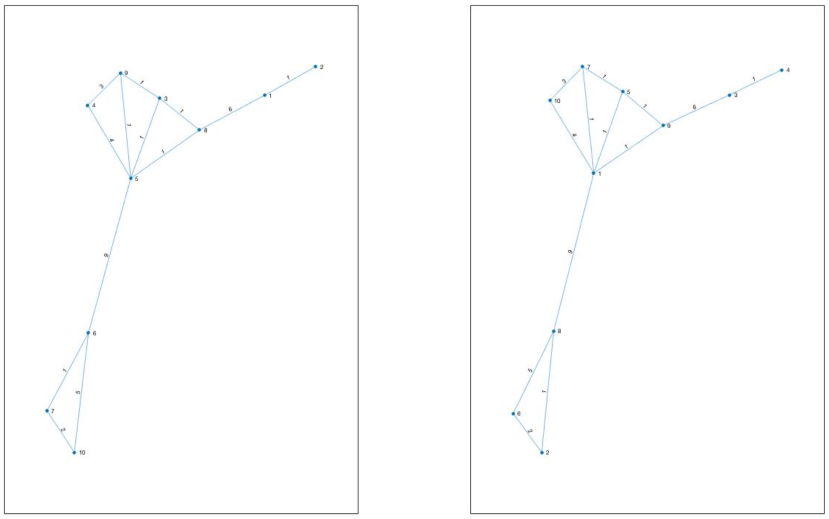

Figure 3: Weighted graphs consisting of 10 nodes

Finally, we do a convergence time analysis of the algorithm with increasing number of vertices. We run the algorithm for N=2 to N=12 nodes for a total of 100 iterations and average the convergence time. Figure 4 presents the convergence time of our algorithm with varying number of nodes/vertices (N).

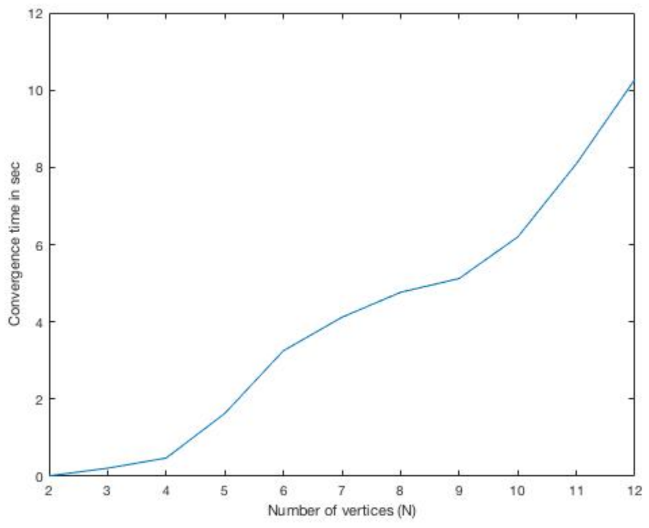

Figure 4: Weighted graphs consisting of 10 nodes

0.7 Conclusion

We have presented a Lagrangian relaxation method for exact graph matching. The algorithm works extremely well for both isomorphic and weighted graph matching. Our approach was to construct a free energy function and minimize it with respect to μ and λ and subsequently maximize it with respect to σ. Please note that [1] suggests Conjugate gradient method for the minimization of cost function in the 'Descent Step'. However because of the increase in the complexity of the gradient and Hessian computation with increasing number of vertices, we have minimized the free energy function using Interior Point Method (by using the fmincon function of MATLAB). The optimal objective value of the initial cost function $||GM - Mg||^2$ should be close to zero for this algorithm to perform reliably. Hence, this algorithm does not work well with noisy isomorphism (inexact matching).

Bibliography

[1] A. Rangarajan and E. Mjolsness, "A lagrangian relaxation network for graph matching," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, p. 1365–1381, 1996.

[2] K. Riesen, X. Jiang, and H. Bunke, *Exact and Inexact Graph Matching: Methodology and Applications*, 02 2010, vol. 40, pp. 217–247.

[3] J. Yan, X.-C. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching," 06 2016, pp. 167–174.

[4] J. Hartmanis, "Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson)," *SIAM Review*, vol. 24, no. 1, p. 90–91, 1982.

[5] F. Zhou and F. D. L. Torre, "Factorized graph matching," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[6] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, p. 377–388, 1996.

[7] J. Yan, Y. Tian, H. Zha, X. Yang, Y. Zhang, and S. M. Chu, "Joint optimization for consistent multiple graph matching," *2013 IEEE International Conference on Computer Vision*, 2013.