Inexact Graph Matching: Applications in Statistical Connectomics

Joshua T. Vogelstein, John M. Conroy, Louis J. Podrazik, Steven G. Kratzer, R. Jacob Vogelstein, and Carey E. Priebe

Abstract—It is becoming increasingly popular to represent myriad and diverse data sets as graphs. When the labels of vertices of these graphs are unavailable, graph matching (GM)—the process of determining whether the graphs are isomorphic to one another—is a computationally daunting problem. This work presents an inexact strategy for GM. Specifically, we relax the feasible region to its convex hull and then apply a well known and efficient nonlinear programming algorithm, Frank-Wolfe, to the objective function. Though this relaxation is convex, the point around which the local approximation is made determines the optimum. We therefore consider a number of initializations based on the geometry of the convex hull. Multiple restarts of this algorithm lead to performance that exceeds the previous state-of-the-art in *all* of 16 benchmark tests. Moreover, this approach is fast, scaling cubically with the number of vertices, requiring only a few minutes on standard modern laptops for graphs with up to a few hundred vertices. We illustrate this approach via a brain-graph ("connectome") application in which vertices represent neurons in a small nematode brain (the *Caenorhabditis elegans* worm), and edges represent either chemical or electrical synapses. For every chemical connectome and several electrical connectomes, this approach found the optimal solution. Although this strategy already natively operates on unweighted and weighted graphs, either directed or undirected, we propose a number of possible extensions, and make all code available.

Index Terms—statistical inference, graph theory, network theory, structural pattern recognition, connectome.

1 Introduction

A graph matching (GM) algorithm is any algorithm whose goal is to "align" any set of $n \geq 2$ graphs such that each vertex in one graph can be "assigned" to its corresponding vertex in the other graphs. Perhaps due to its complex computational properties (it is $\mathcal{NP}$-hard [1]), GM has received widespread attention in both the mathematical graph theory and computer science communities [2]. Moreover, the potential span of applications of graph matching algorithms is vast, ranging from neural coding [3] to machine vision [4].

Our motivation for this work includes the bourgeoning field called "connectomics": the study of brain-graphs. In brain-graphs, vertices represent (collections of) neurons and edges represent either functional dependencies or structural connections [5]. In some scenarios vertices, are labeled. For example, when vertices represent single neurons in invertebrates [6], or macro-anatomical gyral regions in vertebrates [7], [8]. However, in other scenarios, even whether vertices can be labeled is questionable. For example, if one desired to compare brain-(sub)graphs from parts of brains across species or within vertebrate organisms, there is no known vertex assignment. In these scenarios GM might be an important element of any statistical analysis of these brain-graphs [9], [10].

We therefore propose a novel inexact graph matching algorithm. The intuition is relatively simple: GM is computationally difficult because the set of feasible solutions is (i) non-differentiable and (ii) multimodal (and large). A common approach to approximating difficult nonlinear programming problems is to relax the constraints on the feasible region. By relaxing the non-differentiable constraint, any gradient based algorithm may be applied to the problem [11]. Unfortunately, the multimodality of the solution space implies that the initialization will, in general, be important. Multiple "principled" restarts can potentially facilitate an efficient stochastic search strategy.

This manuscript describes an algorithm that approximately solves graph matching in cubic time (with very small leading constants). Via numerical experiments, we demonstrate that this approach outperforms several state-of-the-art algorithms on all tests in a standard benchmark library [12], indicating both its efficiency and its effectivity. We then test this approach on a brain-graph matching problem: matching the brain-graphs of a small nematode with 302 vertices. We are able to find the optimal solution after 3 restarts for each randomly permuted example. We are therefore optimistic that this algorithm will be useful for the massive graphs ($\mathcal{O}(10^5)$ vertices) promised to arise due to various ongoing con-

- *J.T. Vogelstein and C.E. Priebe are with the Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218. E-mail: {joshuav,cep}@jhu.edu, {conroyjohnm,ljpodra,sgkratz}@gmail.com, jacob.vogelstein@jhuapl.edu*
- *J.M. Conroy, L.J. Podrazik and S.G. Kratzer are with Institute for Defense Analyses, Center for Computing Sciences, Bowie, MD 20708.*
- *R.J. Vogelstein is with the Johns Hopkins University Applied Physics Laboratory, Laurel, MD, 20723.*

This work was partially supported by the Research Program in Applied Neuroscience.

nectome projects [13], [14].

2 METHODS

2.1 Preliminaries

A labeled graph $G = (\mathcal{V}, \mathcal{E})$ consists of a vertex set, $\mathcal{V} = [n] = \{1, \ldots, n\}$, where n is number of vertices and an edge set $\mathcal{E} \subseteq n^2$. Let A be the adjacency matrix representations of graph such that $A_{uv} = 1$ if there is an edge from u to v, and $A_{uv} = 0$ otherwise. Note that the below follows for directed/undirected and loopy/non-loopy graphs. Let $\mathcal{Q}$ be the set of permutation matrices, that is, matrices with a single one in each row and column, and zeros otherwise. Given a pair of adjacency matrices, A and B, to graph match A with B is to find a permutation matrix Q such that $QAQ^\mathsf{T} = B$. In this work, we propose a novel inexact graph matching algorithm, essentially a Frank-Wolfe algorithm with multiple restarts. We demonstrate the efficacy of this algorithm over the previous state-of-the-art on a reference library of benchmarks.

2.2 Assignment Problems

2.3 A QAP Approach to GM

Graph matching can be cast as a quadratic assignment problem [15]

$$Q_{AB} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| QAQ^\mathsf{T} - B \right\|_F^2, \tag{1}$$

where the permutation matrix Q_{AB} induces a labeling of the vertices of A onto those of B. A bit of linear algebra simplifies Eq. (1) to give

$$\underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| QAQ^\mathsf{T} - B \right\|_F^2 = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} -tr(B^\mathsf{T} QAQ^\mathsf{T}), \tag{2}$$

which follows from the definition of the Frobenius norm and canceling appropriately. Note that the objective function on the right-hand-side is equivalent to the standard representation of the quadratic assignment problem (QAP) [2]:

$$\hat{\sigma} = \underset{\sigma}{\operatorname{argmin}} \sum_{u,v \in \mathcal{V}} a_{\sigma(u),\sigma(v)} b_{uv} = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \sum_{u,v \in \mathcal{V}} q_{uv} a_{uv} q_{vu} b_{uv} \tag{3}$$

where σ is a permutation function, $\sigma : [n] \mapsto [n]$. Unfortunately, QAP is an $\mathcal{NP}$-complete problem [1]. The primary difficulty in solving Eq. (1) is the discrete non-convex constraint set. Thus, it is natural to consider an approximation with the constraints relaxed. Since the convex hull of permutation matrices is the set of doubly stochastic matrices (matrices whose rows and columns all sum to one), we define the approximate quadratic assignment problem to be

$$\widetilde{Q}_{AB} = \underset{Q \in \mathcal{D}}{\operatorname{argmin}} \left\| QAQ^\mathsf{T} - B \right\|_F^2, \tag{4}$$

where $\mathcal{D}$ is the set of doubly stochastic matrices. When the permutation matrix constraint is relaxed, the equivalence shown in Eq. (2) no longer holds. Thus, the solution to this relaxed problem is not necessarily a permutation matrix; even if it is, there is no guarantee that it is the same permutation matrix that satisfies the original permutation-constrained problem. Nonetheless, we proceed by attempting to solve

$$\hat{\widetilde{Q}}_{AB} \approx \underset{Q \in \mathcal{D}}{\operatorname{argmin}} -tr(B^\mathsf{T} QAQ^\mathsf{T}), \tag{5}$$

considering it an auxiliary function for which we can compute gradients and ascend a likelihood, unlike the permutation-constrained case.

The Frank-Wolfe (FW) algorithm is a successive linear programming algorithm originally designed for solving quadratic problems with linear (equality and/or inequality) constraints [16]. It later became used more generally for nonlinear programming problems [17]. Let $f(Q) = -tr(B^\mathsf{T} QAQ^\mathsf{T})$ be our objective function. Unfortunately, this objective function is not necessarily convexx. This follows from computing the Hessian of f with respect to Q

$$\nabla_Q^2 = B \otimes A + B^\mathsf{T} \otimes A^\mathsf{T}, \tag{6}$$

which is not necessarily positive definite ($\otimes$ indicates the Kronecker product). This means that it will potentially be multi-modal, making initialization important. We therefore develop a nonlinear programing algorithm for approximately solving Eq. (1) using multiple restarts. Below we provide details and explanation of each step.

Step 0: Choose an initial estimate While any doubly stochastic matrix would be a feasible initial point, two choices seem natural: (i) the "flat doubly stochastic matrix," $J = \mathbf{1}\mathbf{1}^\mathsf{T}/n$, which is the center of the feasible region, and (ii) the identity matrix, which is a permutation matrix. Therefore, if we run `QAP` once, we always start with one of those two. If we use multiple restarts, each initial point is "near" the flat matrix. Specifically, we sample K, a random doubly stochastic matrix using 10 iterations of Sinkhorn balancing [18], and let $Q^{(0)} = (J + K)/2$. Given this initial estimate, we iterate the following five steps until convergence.

Step 1: Compute the gradient The gradient of f with respect to Q is given by

$$\nabla_Q^{(j)} = \partial f / \partial Q^{(j)} = -AQ^{(j)} B^\mathsf{T} - A^\mathsf{T} Q^{(j)} B. \tag{7}$$

Step 2: Find the closest doubly stochastic matrix Instead of directly descending this gradient, we search for the direction of the doubly stochastic matrix closest to this gradient. Noting that that direction may be computed by the dot-product operator, we have

$$L^{(j)} = \underset{L \in \mathcal{D}}{\operatorname{argmin}} \langle L, \nabla_Q^{(j)} \rangle. \tag{8}$$

Eq. (8) can be solved as a Linear Assignment Problem (LAP). More specifically, a LAP can be written as

$$L_{AB} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| QA - B \right\|_F^2. \tag{9}$$

When B is the identity matrix I, the above can be simplified as

$$\begin{aligned} L_{AI} &= \operatorname*{argmin}_{Q\in\mathcal{Q}} \|QA - I\|_F^2 = \operatorname*{argmin}_{Q\in\mathcal{Q}} (QA-I)^\mathsf{T}(QA-I) \\ &= \operatorname*{argmin}_{Q\in\mathcal{Q}} tr(A^\mathsf{T}Q^\mathsf{T}QA) - tr(2QA) - tr(II) \\ &= \operatorname*{argmin}_{Q\in\mathcal{Q}} -\langle Q, A\rangle. \end{aligned} \tag{10}$$

In other words, letting $B = I$, the projection of a matrix onto its nearest doubly stochastic matrix is a LAP problem. While Eq. (10) cannot be solved directly, as above, we can relax the permutation matrix constraint to the doubly stochastic matrix constraint

$$\widetilde{L}_{AI} = \operatorname*{argmin}_{Q\in\mathcal{D}} -\langle Q, A\rangle. \tag{11}$$

Since the permutation matrices are the extremal points of the set of doubly stochastic matrices, finding the minimum of Eq. (11) is guaranteed to yield a permutation matrix (as minima are necessarily at the vertices). Thus, letting $A = \nabla_Q^{(j)}$, solving Eq. (11)—which is a linear problem with linear and non-negative constraints—is equivalent to solving Eq. (8). Fortunately, the Hungarian algorithm solves any LAP in $\mathcal{O}(n^3)$ [15], thus this projection is relatively computationally efficient.[1]

Step 3: Update the direction Given $L^{(j)}$, the new direction is given by

$$d^{(j)} = L^{(j)} - Q^{(j)}. \tag{12}$$

Step 4: Line search Given this direction, one can then perform a line search to find the doubly stochastic matrix that minimizes the objective function along that direction:

$$\alpha^{(j)} = \operatorname*{argmin}_{\alpha\in[0,1]} f(Q^{(j)} + \alpha^{(j)} d^{(j)}). \tag{13}$$

This can be performed exactly, because f is a quadratic function.

Step 5: Update Q Finally, the new estimated doubly stochastic matrix is given by

$$Q^{(j+1)} = Q^{(j)} + \alpha^{(j)} d^{(j)}. \tag{14}$$

Stopping criteria Steps 1–5 are iterated until convergence, computational budget limits, or some other stopping criterion is met. These 5 steps collectively comprise the FW algorithm for solving Eq. (5). Note that while $Q^{(j)}$ will generally not be a permutation matrix, we do not project $Q^{(j)}$ back onto the set of permutation matrices between each iteration, as that projection requires $\mathcal{O}(n^3)$ time.

Projecting onto the set of permutation matrices Let $Q^{(J+1)}$ be the doubly stochastic matrix resulting from the final iteration. We project $Q^{(J+1)}$ onto the set of permutation matrices, yielding

$$\hat{Q} = \operatorname*{argmin}_{Q\in\mathcal{Q}} -\langle Q^{(J+1)}, Q\rangle, \tag{15}$$

which is a LAP, and yields an approximate solution to the original QAP. Let FW appended with a projection onto the permutation matrices be denoted by `QAP`.

Multiple restarts We refer to multiple re-starts of `QAP` with subscripts; that is, the performance of $\texttt{QAP}_m$ is the best result of m pseudo-random re-starts of `QAP`. We continue restarting until either we converge to a known global solution or we exceed our computational budget. Note that `QAP` natively operates on matrices, which could correspond to either weighted or unweighted, directed or undirected graphs.

3 Numerical Results

3.1 QAP benchmarks

We first compare the performance of $\texttt{QAP}_m$ with recent state-of-the-art approaches on the QAP benchmark library [12]. Specifically, [19] reported improved performance in all but two cases, in which the QPB method of Cremers et al. [20] achieved a lower minimum. We compare $\texttt{QAP}_m$ with the previous state-of-the-art algorithm. In *all* cases, $\texttt{QAP}_3$ outperforms the previous best result, often by orders of magnitude in terms of relative error. In three cases, $\texttt{QAP}_{100}$ achieves the absolute minimum. In 12 out of 16 cases, the simple $\texttt{QAP}_1$ algorithm outperforms the others (starting with the flat doubly stochastic matrix). See Figure 1 for quantitative comparisons.

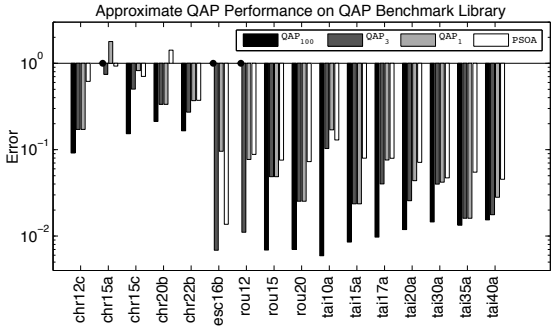

Fig. 1: $\texttt{QAP}_3$ outperforms the previous state-of-the-art (PSOA) on all 16 benchmark graph matching problems. Moreover, $\texttt{QAP}_1$ outperforms PSOA on 12 of 16 tests. For 3 of 16 tests, $\texttt{QAP}_{100}$ achieves the minimum (none of the other algorithms ever find the absolute minimum), as indicated by a black dot. Let f_* be the minimum and $\hat{f}_x$ be the minimum achieved by algorithm x. Error is $\hat{f}_x/f_* - 1$.

3.2 LAP vs. QAP

Much like the QAP objective function from Eq. (1) can be simplified to Eq. (2), the LAP objective function can

1. More efficient algorithms are available for certain special cases, for example, whenever the matrix-vector multiplication operation is fast (for example, when both A and B are sparse).

be similarly simplified, to give

$$Q_{LAP} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \|QA - B\|_F^2 = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} tr(QAB^\mathsf{T}). \quad (16)$$

The gradient of the argument on the right-hand-side of the above equation is $2AB^\mathsf{T}$. Comparing this gradient to that of the QAP objective function—Eq. (7)—one can see that when (i) $Q^{(j)}$ is the identity matrix and (ii) both A and B are symmetric (for example, for undirected graphs), the two gradients are identical. Thus, if QAP is initialized at the identity matrix and the graphs are undirected, the first permutation matrix—the output of Step 2—is identical to $\hat{Q}_{LAP}$; although the line search will make $Q^{(1)} \neq Q_{LAP}$, in general.

3.3 Algorithm Complexity and leading constants

Both GM and its closely related counterpart, graph isomorphism (GI), are computationally difficult. There exist no known algorithms for which worst case behavior is polynomial [21]. While GM is known to be $\mathcal{NP}$-hard, it remains unclear whether GI is in $\mathcal{P}$, $\mathcal{NP}$, or its own intermediate complexity class, $\mathcal{NP}$-isomorphism (or isomorphism-complete). Yet, for large classes of GI and GM problems, linear or polynomial time algorithms are available [22]. Moreover, at worst, it is clear that GI is only "moderately exponential," for example, $\mathcal{O}(\exp\{n^{1/2+o(1)}\})$ [23]. Unfortunately, even when linear or polynomial time GM or GI algorithms are available for special cases of graphs, the constants are typically unbearably large. For example, if all graphs have degree less than k, there is a linear time algorithm for GI. However, the hidden constant in this algorithm is $512k^3!$ [24]. We therefore determined the average complexity of our algorithm *and* the leading constants. Figure 2 suggests that our algorithm is not just cubic in time, but also has very small leading constants ($\approx 10^{-6}$ seconds), making using this algorithm feasible for even reasonably large graphs.

3.4 Brain-Graph Matching

A "connectome" is a brain-graph in which vertices correspond to (collections of) neurons, and edges correspond to connections between them. The *Caenorhabditis elegans* (*C. elegans*) is a small worm (nematode) with 302 labeled vertices. We consider the subgraph with 279 somatic neurons that form edges with other neurons [6], [25]. Two distinct kinds of edges exist between vertices: chemical and electrical "synapses" (edges). Any pair of vertices may have several edges of each type. Moreover, some of the synapses are hyperedges amongst more than two vertices. Thus, the connectome of a *C. elegans* may be thought of as a weighted multi-hypergraph, where the weights are the number of edges of each type. The $\texttt{QAP}_m$ algorithm natively operates on weighted or unweighted graphs. We therefore conducted the following synthetic experiments. Let $A_{uvz} \in \{0, 1, 2, \ldots\}$ be the number of synapses from neuron v to neuron u of type z (either chemical or electrical), and let $A_z = \{A_{uv}\}_{u,v \in [279]}$ for $z \in \{e, c\}$ corresponding to the electrical or chemical connectome. Let $B_{iz} = Q_{iz} A_z Q_{iz}^\mathsf{T}$, for some Q_{iz} chosen uniformly at random from $\mathcal{Q}$ for $i = 1, \ldots, s$. For each i and z, obtain $\hat{Q}_{iz}$ using $\texttt{QAP}_m$ as described above. Define "accuracy" as $\frac{1}{279} \sum_{uv} Q_{iz} \hat{Q}_{iz}$. Table 1 shows some summary results of applying $\texttt{QAP}_m$ to both A_c and A_e for $s = 10$ times. Note that average solution time is actually smaller than predicted via simulations. Further note that while the electrical connectome was more difficult, the median number of restarts was less than 30. Our stopping criteria on the number of restarts was either (i) perfect assignment or (ii) 30 restarts. Therefore, this approach achieved perfect assignment sometimes even on this harder assignment problem.

To investigate the performance of QAP using an undirected graph model, we repeated the above analysis using binarized symmeterized versions of the graphs ($A_{uvz} = 1$ if and only if $A_{uvz} \geq 1$ or $A_{vuz} \geq 1$). The resulting summary statistics are nearly identical to those presented in Table 1, although runtime was less than half the time.

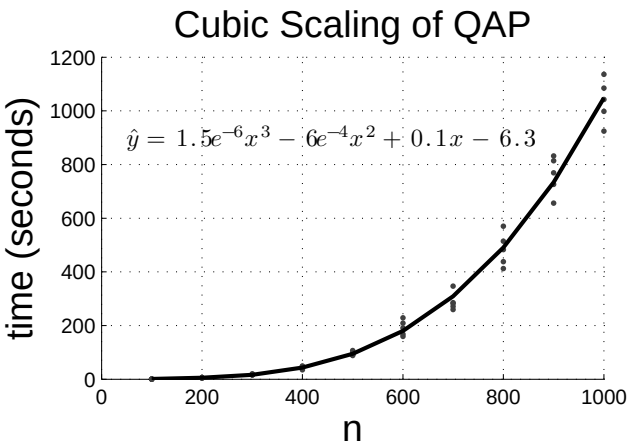

Fig. 2: Performance of QAP as function of number of vertices. Data was sampled from an Erdös-Rényi model with $p = log(n)/n$. Each dot represents a single simulation. The solid line is the best fit cubic function. Note the leading constant is 10^{-6} seconds.

TABLE 1: Brain-graph matching summary statistics for both the chemical and electrical connectome. The maximum number of restarts for both was 30. The median number of restarts for the electrical connectome is less than 30, implying that our approach found the optimal solution many times, just not always.

	chemical	electrical
Median Accuracy	100%	59.5%
Median # Restarts	3	26
Avg. Solution Time	182 sec.	226 sec.

4 DISCUSSION

This work presents an inexact graph matching algorithm based on the Frank-Wolfe algorithm. While others have incorporated the FW algorithm as a subroutine of a graph matching strategy [19], we modified the FW algorithm for GM in a few ways. First, after FW is finished, we project the resulting doubly stochastic matrix onto the set of permutation matrices. Second, we initialize the algorithm using either the identity matrix or the doubly flat matrix (the matrix where all elements are $1/n$). These choices seem to us to be the most obvious places to start if one must choose. Third, if one of those choices does not work, we restart FW with other "nearby" initial points. These modifications facilitate improved performance on *all* the benchmarks we considered. Moreover, although the algorithm scales cubically with the number of vertices, the leading constants are very small ($\mathcal{O}(10^{-6})$ seconds), so the algorithm runs quite fast on reasonably sized networks (e.g., $n \approx 100$). Indeed, on a biologically inspired GM problem, *C. elegans* connectome mapping, this approach was both fast and effective.

Unfortunately, even with very small leading constants for this algorithm, as n increases, the computational burden gets quite high. For example, extrapolating the curve of Figure 2, this algorithm would take about 2 years to finish (on a standard laptop from 2009) when $n = 20,000$. We hope to be able to perform GM on graphs much larger than that, given that the number of neurons in even a fly brain, for example, is $\mathcal{O}(10^5)$. Therefore, more efficient implementations are of interest.

Although $\texttt{QAP}_n$ consistently found the optimal solution for the *C. elegans* chemical connectome, connectomes for different organisms even within a species are unlikely to be identical. Even if all connectomes of a particular species were identical, measurement error will likely persist [26]. Therefore, $\texttt{QAP}_n$'s scientific utility will largely rest on its performance under noisy conditions, which we aim to explore in future work.

Additional future work might generalize $\texttt{QAP}_n$ in a number of ways. First, that QAP and LAP are so similar suggests that perhaps one could simply implement a single iteration of QAP starting from the identity. While not changing the order of complexity, it could reduce computational time by at least an order of magnitude, without drastically changing performance properties (because convergence typically requires around $5 - 15$ iterations for the graphs we tested). The relative performance/computational cost trade-off merits further theoretical investigations. Second, the most "costly" subroutine is LAP. Fortunately, LAP is a quadratic optimization problem with linear constraints. A number of parallelized optimization strategies could therefore potentially be brought to bear on this problem [27]. Third, for brain-graphs, we have some prior information that could easily be incorporated in the form of vertex attributes. For example, position in the brain, cell type, etc., could be used to measure "dissimilarity" between vertices. The objective function could then be modified to give

$$\widetilde{Q}_{AB} = \underset{Q \in \mathcal{D}}{\operatorname{argmin}} \left\| QAQ^{\mathsf{T}} - B \right\|_F^2 + \lambda J(Q), \qquad (17)$$

where $J(Q)$ is a dissimilarity based penalty and λ is a hyper-parameter. Finally, although this approach natively operates on both unweighted and weighted graphs, multi-graphs are a possible extension.

In conclusion, this manuscript has presented a GM algorithm that is fast, effective, and easily generalizable. To facilitate further development and applications, all the code and data used in this manuscript is available from the first author's website, http://jovo.me.

ACKNOWLEDGMENTS

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[2] D. Conte, P. Foggia, C. Sansone, and M. Vento, "THIRTY YEARS OF GRAPH MATCHING," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 265–298, 2004.

[3] J. Richiardi, H. Eryilmaz, S. Schwartz, P. Vuilleumier, and D. Van De Ville, "Decoding brain states from fMRI connectivity graphs." *NeuroImage*, Jun. 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/20541019

[4] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, Jul. 1997. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=598235

[5] O. Sporns, *Networks of the Brain*. The MIT Press, 2010. [Online]. Available: http://www.amazon.com/Networks-Brain-Olaf-Sporns/dp/0262014696

[6] J. White, E. Southgate, J. N. Thomson, and S. Brenner, "The structure of the nervous system of the nematode caenorhabditis elegans." *Philosophical Transactions of Royal Society London. Series B, Biological Sciences*, vol. 314, no. 1165, pp. 1–340, 1986.

[7] B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe, A.-M. Dogonowski, M. Ernst, D. Fair, M. Hampson, M. J. Hoptman, J. S. Hyde, V. J. Kiviniemi, R. Kötter, S.-J. Li, C.-P. Lin, M. J. Lowe, C. Mackay, D. J. Madden, K. H. Madsen, D. S. Margulies, H. S. Mayberg, K. McMahon, C. S. Monk, S. H. Mostofsky, B. J. Nagel, J. J. Pekar, S. J. Peltier, S. E. Petersen, V. Riedl, S. a. R. B. Rombouts, B. Rypma, B. L. Schlaggar, S. Schmidt, R. D. Seidler, G. J. Siegle, C. Sorg, G.-J. Teng, J. Veijola, A. Villringer, M. Walter, L. Wang, X.-C. Weng, S. Whitfield-Gabrieli, P. Williamson, C. Windischberger, Y.-F. Zang, H.-Y. Zhang, F. X. Castellanos, and M. P. Milham, "Toward discovery science of human brain function." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 10, pp. 4734–9, Mar. 2010. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2842060\&tool=pmcentrez\&rendertype=abstract

[8] E. T. Bullmore and D. S. Bassett, "Brain Graphs: Graphical Models of the Human Brain Connectome." *Annual review of clinical psychology*, no. November 2010, Apr. 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21128784

[9] J. T. Vogelstein, W. R. Gray, R. J. Vogelstein, and C. E. Priebe, "Graph Classification using Signal Subgraphs: Applications in Statistical Connectomics," *Submitted for publication*, 2011.

[10] J. T. Vogelstein and C. E. Priebe, "Bayes Optimal Graph Classification: Applications in Statistical Connectomics," *Submitted for publication*, 2011.

[11] O. L. Mangasarian, *Nonlinear Programming (Classics in Applied Mathematics)*. Society for Industrial Mathematics, 1987. [Online]. Available: http://www.amazon.com/Nonlinear-Programming-Classics-Applied-Mathematics/dp/0898713412

[12] R. E. Burkard, S. E. Karisch, and F. Rendl, "QAPLIB A Quadratic Assignment Problem Library," *Journal of Global Optimization*, vol. 10, no. 4, pp. 391–403, 1997. [Online]. Available: http://www.springerlink.com/content/n5468q3g33184443/fulltext.pdf
[13] "Human Connectome Project - NIH Neuroscience Blueprint." [Online]. Available: http://humanconnectome.org/consortia/
[14] "Open Connectome Project." [Online]. Available: http://openconnectomeproject.org/
[15] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. SIAM, 2009. [Online]. Available: http://books.google.com/books?id=nHIzbApLOr0C\&pgis=1
[16] M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956. [Online]. Available: http://www.ams.org/mathscinet/search/publications.html?pg1=MR\&s1=MR0089102
[17] S. P. Bradley, A. C. Hax, and T. L. Magnanti, *Applied Mathematical Programming*. Addison-Wesley, 1977. [Online]. Available: http://www.amazon.com/Applied-Mathematical-Programming-Stephen-Bradley/dp/020100464X
[18] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 876–879, 1964. [Online]. Available: http://www.jstor.org/stable/2238545
[19] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2227–2242, 2009. [Online]. Available: http://eprints.pascal-network.org/archive/00004435/
[20] C. Schellewald, S. Roth, and C. Schnörr, "Evaluation of Convex Optimization Techniques for the Weighted Graph-Matching Problem in Computer Vision," in *Proceedings of the 23rd DAGMSymposium on Pattern Recognition*. Springer-Verlag, 2001, pp. 361–368.
[21] S. Fortin, "The Graph Isomorphism Problem," *Technical Report, University of Alberta, Dept of CS*, 1996. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.9419
[22] L. Babai, P. Erds, and S. M. Selkow, "RANDOM GRAPH ISOMORPHISM * 1 . A straightforward algorithm . The problem of testing graphs for isomorphism belongs to those combinatorial search problems for which no polynomial-time algorithm is available as yet . It is , however , striking , that even t," *Society*, no. August, 1980.
[23] L. Babai, "Moderately Exponential Bound for Graph Isomorphism," pp. 34–50, Aug. 1981. [Online]. Available: http://portal.acm.org/citation.cfm?id=647890.739270
[24] J. Chen, "A Linear-Time Algorithm for Isomorphism of Graphs of Bounded Average Genus," *SIAM Journal on Discrete Mathematics*, vol. 7, no. 4, p. 614, Nov. 1994. [Online]. Available: http://portal.acm.org/citation.cfm?id=197033.197062
[25] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii, "Structural Properties of the Caenorhabditis elegans Neuronal Network," *PLoS Computational Biology*, vol. 7, no. 2, p. e1001066, Feb. 2011.
[26] M. Helmstaedter, K. L. Briggman, and W. Denk, "High-accuracy neurite reconstruction for high-throughput neuroanatomy," *Nature Neuroscience*, vol. 14, no. 8, pp. 1081–1088, Jul. 2011. [Online]. Available: http://dx.doi.org/10.1038/nn.2868
[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Foundations and Trends in Machine Learning," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1—-122, 2011. [Online]. Available: http://www.nowpublishers.com/product.aspx?product=MAL\&doi=2200000016

Lou Podrazik

Steve Kratzer

R. Jacob Vogelstein

Carey E. Priebe

Joshua T. Vogelstein

John C. Conroy