(Brain) Graph Matching via Fast Approximate Quadratic Programming

JoVo, cep

July 20, 2012

Abstract

Graph matching (GM)—the process of finding an optimal permutation of the vertices of one graph to minimize adjacency disagreements with the vertices of another—is rapidly becoming an increasingly important computational problem, arising in fields ranging from machine vision to chemical engineering to neuroscience. Because GM is NP-hard, exact algorithms are unsuitable for today's massive graphs; yet, scalable GM algorithms have received short shrift. GM can be formulated as a quadratic program with linear and binary constraints. We develop a fast approximate quadratic (`FAQ`) assignment algorithm to approximately solve a relaxed quadratic program with only linear constraints. `FAQ` scales cubicly with the number of vertices, and demonstrates marked improvements over previous state-of-the-art on nearly all benchmarks. Moreover, our non-convex formulation facilitates multiple restarts; $2-3$ wisely chosen initial conditions yield the best objective function on *all* benchmarks. We find qualitatively similar results for our motivating application: brain-graph matching. Unfortunately, the computational complexity of `FAQ` scales too poorly to use it for mammalian brain-graphs, with millions or billions of vertices. To inspire further development of approximate solutions to these problems, this work is available from on the first author's website, `http://jovo.me`.

1 Introduction

Graph matching—the process of finding an optimal permutation of the vertices of one graph to minimize adjacency disagreements with the vertices of another—is a famously computationally daunting problem (see, for example, "Thirty Years of Graph Matching in Pattern Recognition" [1]). Specifically, graph matching is an $\mathcal{NP}$-hard problem, in particular, we do not know whether a polynomial time algorithm can solve it [2]. Perhaps the most prominent special case of graph matching is the traveling salesman problem [3]. As such, performance of graph matching algorithms are usually evaluated on graphs with $\dot{\approx}10$ vertices, or at maximum $\dot{\approx}100$ (see [4] for a description of the standard set of benchmarks). Yet, it is increasingly popular to represent large data sets by a graph, and thus increasingly desirable to consider matching large graphs.

The motivating application for this work is *brain-graph matching*. A brain-graph (aka, a connectome) is a graph for which vertices represent (collections of) neurons and edges represent connections between them [5, 6]. Via Magnetic resonance (MR) imaging, one can image the whole brain and estimate connectivity across voxels, yielding a voxelwise connectome with up to $\dot{\approx}10^6$ vertices and $\dot{\approx}10^9$ edges [7]. Comparing brains is an important step for many neurobiological inference tasks. For example, it is becoming increasingly popular to diagnose neurological diseases via comparing brain images [8]. To date, however, these comparisons have largely rested on anatomical (e.g., shape) comparisons, not graph comparisons. This is despite the widely held doctrine that many psychiatric disorders are fundamentally "connectopathies", that is, disorders of the connections of the brain [9, 10, 11, 12]. Currently available tests for connectopic explanation of psychiatric disorders hedge upon first choosing some number of graph invariants to compare across populations. The graph invariant approach to classifying is both theoretically and practically inferior to comparing whole graphs via matching [13].

More generally, state-of-the-art inference procedures for essentially any decision-theoretic or inference task follow from constructing interpoint dissimilarity matrices [14]. Thus, we believe that graph matching of large graphs will become a fundamental subroutine of many statistical inference pipelines operating on graphs. Because the number of vertices of these graphs is so large, exact matching is intractable. Instead, we require inexact matching algorithms (also called "heuristics") that will scale polynomially or even linear [1]. We develop an approach to graph matching based on a relaxation of the quadratic programming problem (QAP). Our approach is only cubic in the number of vertices, and outperforms previously proposed approximate graph matching heuristics on a wide range of benchmark datasets as well as our motivating application.

2 Graph Matching

A labeled graph $G = (\mathcal{V}, \mathcal{E})$ consists of a vertex set $\mathcal{V}$, where $|\mathcal{V}| = n$ is number of vertices, and an edge set $\mathcal{E}$. Note that we are not restricting our formulation to be directed or exclude self-loops. Given a pair of graphs, $G_A = (\mathcal{V}_A, \mathcal{E}_A)$ and $G_B = (\mathcal{V}_B, \mathcal{E}_B)$, where $|\mathcal{V}_A| = |\mathcal{V}_B| = n$, let Π be the set of permutation functions (bijections), $\Pi = \{\pi \colon \mathcal{V}_A \to \mathcal{V}_B\}$. Now consider the following two closely related problems:

- **Graph Isomorphism (GI):** Does there exist a $\pi \in \Pi$ such that $(u, v) \in \mathcal{E}_A$ if and only if $(\pi(u), \pi(v)) \in \mathcal{E}_B$.
- **Graph Matching (GM):** Which $\pi \in \Pi$ minimizes adjacency disagreements between $\mathcal{E}_A$ and the permuted $\mathcal{E}_B$?

Both GI and GM are computationally difficult. GM is at least as hard as GI, since solving GM also solves GI, but not vice versa. It is not known whether GI is in complexity class $\mathcal{P}$ [15]. In fact, GI is one of the few problems for which, if $\mathcal{P} \neq \mathcal{NP}$, then GI might reside in an intermediate complexity class called $\mathcal{GI}$-complete. GM, however, is known to be $\mathcal{NP}$-hard. Yet, for large classes of GI and GM problems, linear or polynomial time algorithms are available [16]. Moreover, at worst, it is clear that GI is only "moderately exponential," for example, $\mathcal{O}(\exp\{n^{1/2+o(1)}\})$ [17]. Unfortunately, even when linear or polynomial time GI or GM algorithms are available for special cases of graphs, the constants are typically unbearably large. For example, if all vertices have degree less than k, there is a linear time algorithm for GI. However, the hidden constant in this algorithm is $512k^3$! (yes, that is a factorial!) [18].

Because we are interested in solving GM for graphs with $\approx 10^6$ or more vertices, exact GM solutions will be computationally intractable. As such, we develop a fast approximate graph matching algorithm. Our approach is based on formulating GM as a quadratic assignment problem (QAP).

3 Graph Matching as a QAP

Graph matching can be formulated as a quadratic assignment problem (QAP). Let $A = (a_{uv}) \in \{0,1\}^{n \times n}$ and $B = (b_{uv}) \in \{0,1\}^{n \times n}$ correspond to the adjacency matrix representations of two graphs that we desire to match. That is, let $a_{uv} = 1$ if and only if $(u, v) \in \mathcal{E}_A$, and similarly for b_{uv}. Moreover, let $\mathcal{P}$ be the set of $n \times n$ *permutation matrices* $\mathcal{P} = \{P : P^\mathsf{T}\mathbf{1} = P\mathbf{1} = \mathbf{1}, P \in \{0,1\}^{n \times n}\}$, where $\mathbf{1}$ is an n-dimensional column vector. We therefore have the following problem:

$$\text{(QAP)} \quad \operatorname*{argmin}_{\pi \in \Pi} \sum_{i,j \in [n]} (a_{ij} - b_{\pi(i)\pi(j)})^2 = \tag{1a}$$

$$\operatorname*{argmin}_{P \in \mathcal{P}} \left\| A - PBP^\mathsf{T} \right\|_F = \tag{1b}$$

$$\operatorname*{argmin}_{P \in \mathcal{P}} tr(A - PBP^\mathsf{T})^\mathsf{T}(A - PBP^\mathsf{T}) = \tag{1c}$$

$$\operatorname*{argmin}_{P \in \mathcal{P}} -tr(BP^\mathsf{T}AP), \tag{1d}$$

where the last equality follows from dropping terms that cancel because P is a permutation matrix. Note that the above algebraic formulation of GM facilitates generalizing the original problem statement. In particular, one can now search for the permutation that minimizes a particular objective function, $f(P) = -tr(BP^\mathsf{T}AP)$. Moreover, it is natural to consider "weighted graph matching" problems, in which each edge is associated with a weight, $a_{uv} \in \mathbb{R}$.

Our approach follows from relaxing the above binary constraints to be non-negative constraints, yielding a quadratic program with *linear* constraints. Thus, the feasible region expands to the convex hull of the permutation matrices: the doubly stochastic matrices, $\mathcal{D} = \{P : P^\mathsf{T}\mathbf{1} = P\mathbf{1} = \mathbf{1}, P \succeq 0\}$, where $\succeq$ indicates an element-wise inequality:

$$\text{(rQAP)} \quad \underset{P}{\text{minimize}} \quad -tr(BP^\mathsf{T}AP) \tag{2a}$$

$$\text{subject to} \quad P \in \mathcal{D}. \tag{2b}$$

rQAP—the above relaxed Quadratic Assignment Problem—is quadratic but not necessarily convex, because the Hessian of its objective function is not necessarily positive definite:

$$\nabla^2 f(P) = -B \otimes A - B^\mathsf{T} \otimes A^\mathsf{T}, \tag{3}$$

where $\otimes$ indicates the Kronecker product. This means that the solution space will potentially be multimodal, making initialization important. With this in mind, below, we describe an algorithm to find a local optimum of rQAP.

4 Fast Approximate Quadratic Assignment Problem Algorithm

Our algorithm, called `FAQ`, has three components:

A. Choose a suitable initial position.

B. Find a local solution to rQAP.

C. Project onto the set of permutation matrices.

Below, we provide details for each component.

A: Find a suitable initial position. While any doubly stochastic matrix would be a feasible initial point, we choose the "flat doubly stochastic matrix," $J = \mathbf{1} \cdot \mathbf{1}^\mathsf{T}/n$, which is the barycenter of the feasible region.

B: Find a local solution to rQAP. As mentioned above, rQAP is a quadratic problem with linear constraints. A number of off-the-shelf algorithms are readily available for finding local optima in such problems. We utilize the Frank-Wolfe algorithm (`FW`), a successive linear programing problem originally devised to solve quadratic problems with linear constraints [19, 20].

Although `FW` is a relatively standard solver, especially as a subroutine for QAP algorithms [21], below we provide a detailed view of applying `FW` to rQAP. Given an initial position, $P^{(0)}$, iterate the following four steps.

Step 1: Compute the gradient $\nabla f(P^{(i)})$: The gradient f with respect to P is given by

$$\nabla f(P^{(i)}) = -AP^{(i)}B^\mathsf{T} - A^\mathsf{T}P^{(i)}B. \tag{4}$$

Step 2: Compute a new putative point $\widetilde{P}^{(i+1)}$: The new putative point is given by the argument that minimizes a first-order Taylor series approximation to $f(P)$ around the current estimate, $P^{(i)}$. The first-order Taylor series approximation to $f(P)$ is given by

$$\widetilde{f}^{(i)}(P) \overset{\triangle}{=} f(P^{(i)}) + \nabla f(P^{(i)})^\mathsf{T}(P - P^{(i)}). \tag{5}$$

Thus, Step 2 of `FW` is

$$\widetilde{P}^{(i+1)} = \underset{P\in\mathcal{D}}{\operatorname{argmin}} f(P^{(i)}) + \nabla f(P^{(i)})^\mathsf{T}(P - P^{(i)}) \tag{6a}$$

$$= \underset{P\in\mathcal{D}}{\operatorname{argmin}} \nabla f(P^{(i)})^\mathsf{T} P. \tag{6b}$$

As it turns out, Eq. (6b) can be solved as a *Linear Assignment Problem* (LAP). The details of LAPs are well known [3], so we relegate them to the appendix. Suffice it to say here, LAPs can be solved via the "Hungarian Algorithm", named after three Hungarian mathematicians [22, 23, 24]. Modern variants of the Hungarian algorithm are cubic in n, that is, $\mathcal{O}(n^3)$, or even faster in the case of sparse or otherwise structured graphs [25, 3]. The $\mathcal{O}(n^3)$ computational complexity of `FW` was the primary motivating factor for utilizing `FW`; generic linear programs can require up to $\mathcal{O}(n^7)$.

Step 3: Compute the step size $\alpha^{(i)}$ Given $\widetilde{P}^{(i+1)}$, the new point is given maximizing the *original* optimization problem, rQAP, along the line segment from $P^{(i)}$ to $\widetilde{P}^{(i+1)}$ in $\mathcal{D}$.

$$\alpha^{(i)} = \underset{\alpha\in[0,1]}{\operatorname{argmin}} f(P^{(i)} + \alpha^{(i)}\widetilde{P}^{(i)}). \tag{7}$$

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

Step 4: Update $P^{(i)}$ Finally, the new estimated doubly stochastic matrix is given by

$$P^{(i+1)} = P^{(i)} + \alpha^{(i)}\widetilde{P}^{(i+1)}. \tag{8}$$

Stopping criteria Steps 1–4 are iterated until some stopping criterion is met (computational budget limits, $P^{(i)}$ stops changing much, or $\nabla f(P^{(i)})$ is close to zero). These four steps collectively comprise the Frank-Wolfe algorithm for solving rQAP.

C: Project onto the set of permutation matrices. Let $\widehat{D}$ be the doubly stochastic matrix resulting from the final iteration of `FW`. We project $\widehat{D}$ onto the set of permutation matrices, yielding

$$\widehat{P} = \underset{P\in\mathcal{P}}{\operatorname{argmin}} -\langle \widehat{D}, P\rangle, \tag{9}$$

where $\langle\cdot,\cdot\rangle$ is the usual Euclidean inner product, i.e., $\langle X, Y\rangle \overset{\triangle}{=} tr(X^\mathsf{T}Y) = \sum_{ij} x_{ij}y_{ij}$. Note that Eq. (9) is a LAP (again, see appendix for details).

5 Results

5.1 Algorithm Complexity and leading constants

As mentioned above, GM is computationally difficult; even those special cases for which polynomial time algorithms are available, the leading constants are intractably large for all but the simplest cases. We therefore determined the average complexity of our algorithm and the leading constants. Figure 1 suggests that our algorithm is not just cubic in time, but also has very small leading constants ($\approx 10^{-9}$ seconds), making using this algorithm feasible for even reasonably large graphs.

5.2 QAP Undirected Benchmarks

We next assess the computational properties of `FAQ` in comparison with other the previous state-of-the-art algorithms. We therefore compare `FAQ` to other approaches using a selection of the QAP benchmark library, QAPLIB [4]. Specifically, [26] created a path following algorithm (`PATH`) based on a convex and concave relaxation of QAP. In that manuscript, the authors considered 16 datasets from the QAPLIB benchmark, the

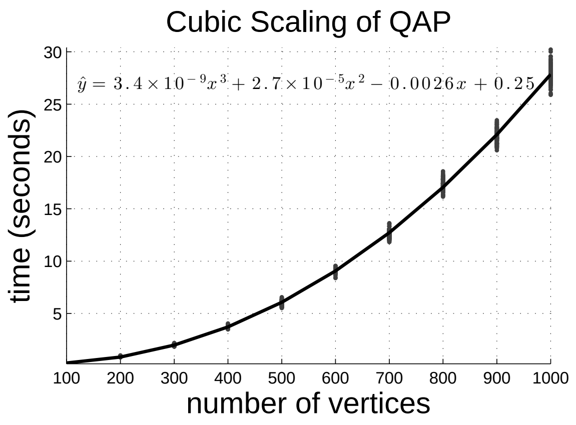

Figure 1: Running time of FAQ as function of number of vertices. Data was sampled from an Erdös-Rényi model with $p = log(n)/n$. Each dot represents a single simulation, with 100 simulations per n. The solid line is the best fit cubic function. Note the leading constant is $\dot{\approx}10^{-9}$ seconds. FAQ finds the optimal objective function value in every simulation.

same 16 datasets as were used in [27], which are known to be "particularly difficult". PATH was shown to outperform other state-of-the-art algorithms on 14 of 16 tests. Specifically, PATH was compared to the Quadratic Programming Bound approach (QGP) of [28], the graduated assignment algorithm (GRAD) of [29], and Umeyama's algorithm (U) [30]. Because either PATH or QBP outperformed GRAD and U on every dataset, Table 1 shows the performance of FAQ versus PATH and QBP. FAQ outperforms both of the previous state-of-the-art cubic algorithms on 13 out of 16 benchmarks. Figure 2 presents the same data graphically. The top panel compares both FAQ and PSOA—which is the minimum of the previous state-of-the-art (either PATH or QBP here)—to the absolute minimum; FAQ get closer than PSOA to the minimum on 13 of 16. The bottom panel shows the ratio of FAQ to PSOA.

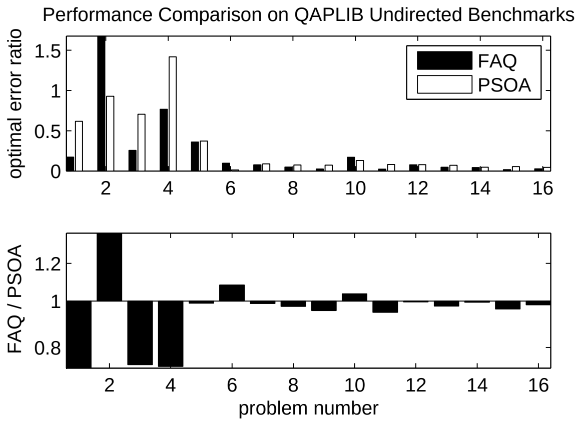

Figure 2: Performance of FAQ relative to the previous state-of-the-art (PSOA) algorithms on the undirected QABLIB benchmarks. Top: The optimal error ratio is defined: $(\hat{f} - f^*)/f^*$, for $\hat{f}$ being the minimum function value found by each algorithm, and f^* is the optimal (minimum) value. Bottom: Ratio of FAQ minimum to PSOA minimum. Note that both panels indicate that FAQ gets closer to the minimum on 13 of 16 benchmarks.

Table 1: Comparison of `FAQ` with the optimal objective function value and previous state-of-the-art on a set of 16 standard benchmarks from QAPLIB. The best (lowest) value is in **bold**. The number of vertices for each problem is the number in its name (second column).

#	Problem	Optimal	`FAQ`	`PATH`	`QBP`
1	chr12c	11156	**13072**	18048	20306
2	chr15a	9896	27584	**19086**	26132
3	chr15c	9504	**11936**	16206	29862
4	chr20b	2298	**3068**	5560	6674
5	chr22b	6194	**8482**	8500	9942
6	esc16b	292	320	300	**296**
7	rou12	235528	**253684**	256320	278834
8	rou15	354210	**371458**	391270	381016
9	rou20	725522	**743884**	778284	804676
10	tai10a	135028	157954	**152534**	165364
11	tai15a	388214	**397376**	419224	455778
12	tai17a	491812	**529134**	530978	550862
13	tai20a	703482	**734276**	753712	799790
14	tai30a	1818146	**1894640**	1903872	1996442
15	tai35a	2422002	**2460940**	2555110	2720986
16	tai40a	3139370	**3227612**	3281830	3529402

5.3 QAP Directed Benchmarks

Nothing in the development of our algorithm depends on the graphs being simple; indeed, `FAQ` applies equally well to directed graphs. To assess the performance of `FAQ` on directed graphs, we compare the performance of our algorithm to the previous state-of-the-art. Liu et al. recently developed an extended path following algorithm for directed graphs [31]. They compare the performance of their algorithm (`EPATH`) with several other algorithms on a set of 16 benchmarks from QAPLIB. In particular, they consider `U` and `GRAD`, as well as an algorithm called `QCV`, which solves a convex relaxation similar to our approach. The `EPATH` algorithm achieves at least as low objective value as the other algorithms on 15 of 16 benchmarks. Our algorithm, `FAQ`, always gets the best of the five algorithms. Table 2 shows the numerical results comparing `FAQ` to `EPATH` and `GRAD`, which sometimes did better than `EPATH`. Note that some of the algorithms achieve the absolute minimum on some benchmarks. Figure 3 compares `FAQ` to whichever other algorithm did best, clearly indicating that `FAQ` is the best on these benchmarks.

5.4 rQAP solves QAP in certain special cases

The above numerical results can be strengthened by the below theoretical results. Note that rQAP relaxes the constraints of Eq. (1d), which suggests that in certain important special cases, the minimum of rQAP will be identical to the minimum of QAP. On the other hand, the equality between Eq. (1c) and Eq. (1d) follows from dropping cross-terms that fall out of the optimization because P is constrained to be a permutation matrix. If we had relaxed the constraints prior to canceling those terms, this equality would not follow. This leads us to wonder in which circumstances are the objective functions of QAP and rQAP equal. The following lemma clarifies:

Lemma 1. *If A and B are the adjacency matrices of simple graphs (symmetric, hollow, and binary) that*

Table 2: Comparison of `FAQ` with optimal objective function value and previous state-of-the-art for undirected graphs. The best (lowest) value is in **bold**. Asterisks indicate achievement of the global minimum. The number of vertices for each problem is the number in its name (second column).

#	Problem	Optimal	`FAQ`	`EPATH`	`GRAD`
1	lipa20a	3683	**3791**	3885	3909
2	lipa20b	27076	**27076***	32081	**27076***
3	lipa30a	13178	**13571**	13577	13668
4	lipa30b	151426	**151426***	**151426***	**151426***
5	lipa40a	31538	**32109**	32247	32590
6	lipa40b	476581	**476581***	**476581***	**476581***
7	lipa50a	62093	**62962**	63339	63730
8	lipa50b	1210244	**1210244***	**1210244***	**1210244***
9	lipa60a	107218	**108488**	109168	109809
10	lipa60b	2520135	**2520135***	**2520135***	**2520135***
11	lipa70a	169755	**171820**	172200	173172
12	lipa70b	4603200	**4603200***	**4603200***	**4603200***
13	lipa80a	253195	**256073**	256601	258218
14	lipa80b	7763962	**7763962***	**7763962***	**7763962***
15	lipa90a	360630	**363937**	365233	366743
16	lipa90b	12490441	**12490441***	**12490441***	**12490441***

are isomorphic to one another, then the minimum of rQAP is equal to the minimum of QAP.

Proof. Because any feasible solution to QAP is also a feasible solution to rQAP, we must only show that the optimal objective function value to rQAP can be no better than the optimal objective function value of QAP. Let $A = PBP^{\mathsf{T}}$, so that $\langle A, PBP^{\mathsf{T}}\rangle = 2m$, where m is the number of edges in A. If rQAP could achieve a lower objective value, then it must be that there exists a $D \in \mathcal{D}$ such that $\langle A, DBD^{\mathsf{T}}\rangle > \langle A, PBP^{\mathsf{T}}\rangle = 2m$ (remember that we are minimizing the negative Euclidean inner product). For that to be the case, it must be that $(DBD^{\mathsf{T}})_{ij} \geq 1$ for some (u, v). That this is not so may be seen by the submultiplicativity of the norm induced by the ℓ_∞ norm: $\|Dx\|_\infty \leq \|D\|_{\infty,\infty} \|x\|_\infty$. Applying this twice (once for each doubly stochastic matrix multiplication) yields our result. $\square$

5.5 Multiple Restarts

Although `FAQ` outperformed `PSOA` on 13 of 16 undirected benchmarks, and always did the best amongst 16 of 16 directed benchmarks, it was annoying to us that we did not do best on all 32 benchmarks. We utilize the non-convexity of rQAP is as a feature, although it can equally well be regarded as a bug (because rQAP is non-convex so the solution found by `FAQ` depends on the initial condition). It is a feature, however, if (i) we have some reason to believe that better solutions exist (many algorithms efficiently compute relatively tight lower bounds [32]), and (ii) we can efficiently search the space of initial conditions. Although we lack any supporting theory of optimality, we do know how to sample feasible starting points. Specifically, we desire that our starting points are "near" the flat matrix, and satisfy the conditions. Therefore, we sample $K \in \mathcal{D}$, a random doubly stochastic matrix using 10 iterations of Sinkhorn balancing [33], and let our initial guess be $P^{(0)} = (J + K)/2$, where J is the doubly flat matrix. We can therefore use any number of restarts with this approach.

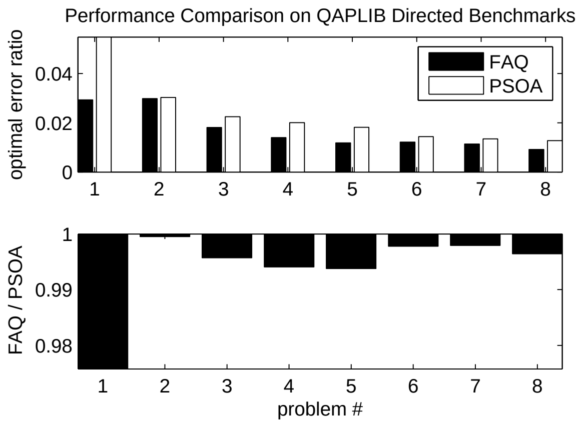

Figure 3: Performance of `FAQ` relative to the previous state-of-the-art (`PSOA`) algorithms on the undirected QABLIB benchmarks. Top and Bottom panels as in Figure 2. Note that `FAQ` gets closer to the minimum on all 8 benchmarks for which the `FAQ` and `PSOA` answer differ.

Table 3 shows the performance of running `FAQ` 3 and 100 times, reporting only the best result (indicated by $\texttt{FAQ}_3$ and $\texttt{FAQ}_{100}$, respectively), and comparing it to the best performing result from Table 1. It only required three restarts to outperform all other cubic algorithms on all 16 of 16 benchmarks. Moreover, after 100 restarts, `FAQ` finds the absolute minimum on 3 of the 16 benchmarks. Figure 4 graphically demonstrates these results. Note that restarting `FAQ` a fixed number of multiple times is still cubic. Future work could investigate performance as a function of the number of restarts.

Table 3: Comparison of `FAQ` with optimal objective function value and the best result from Table 1 on undirected benchmarks. Note that `FAQ` restarted 100 times finds the optimal objective function value in 3 of 16 benchmarks, and that `FAQ` restarted 3 times finds a minimum better than the PSOA on all 16 benchmarks.

#	Problem	Optimal	$\texttt{FAQ}_{100}$	$\texttt{FAQ}_3$	min(`FAQ` ,`PSOA`)
1	chr12c	11156	**12176**	13072	13072
2	chr15a	9896	**9896***	17272	19086
3	chr15c	9504	**10960**	14274	16206
4	chr20b	2298	**2786**	3068	3068
5	chr22b	6194	**7218**	7876	8482
6	esc16b	292	**292***	294	296
7	rou12	235528	**235528***	238134	253684
8	rou15	354210	**356654**	371458	371458
9	rou20	725522	**730614**	743884	743884
10	tai10a	135028	**135828**	148970	152534
11	tai15a	388214	**391522**	397376	397376
12	tai17a	491812	**496598**	511574	529134
13	tai20a	703482	**711840**	721540	734276
14	tai30a	1818146	**1844636**	1890738	1894640
15	tai35a	2422002	**2454292**	2460940	2460940
16	tai40a	3139370	**3187738**	3194826	3227612

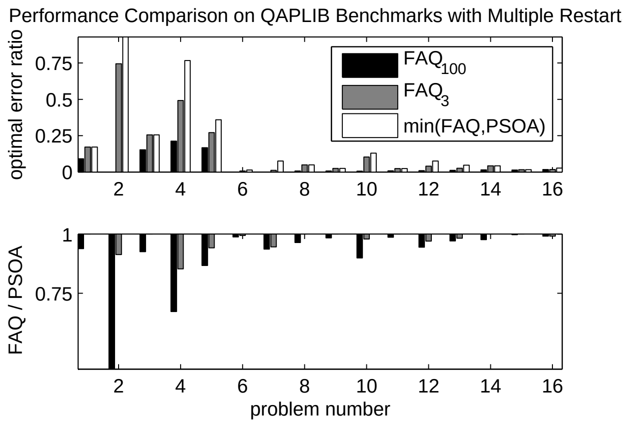

Figure 4: Performance of `FAQ` with multiple restarts on the undirected benchmarks. $\texttt{FAQ}_3$ yields a lower objective function value than the best result from Figure 2, and $\texttt{FAQ}_{100}$ finds the absolute optimal permutation on 3 of the 16 benchmarks. Note that no other algorithm compared ever found the optimal for any of the benchmarks.

5.6 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 [34, 35]. 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 hyper-edges 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. `FAQ` natively operates on weighted or unweighted graphs. We therefore conducted the following synthetic experiments.

Let $A_{ij;z} \in \{0, 1, 2, \ldots\}$ be the number of synapses from neuron i to neuron j of type z (either chemical c or electrical e), and let $A_z = \{A_{ij;z}\}_{i,j\in[279]}$ for $z \in \{e, c\}$ correspond to the electrical or chemical connectome. To generate synthetic data, we let $B_z^{(k)} = Q_z^{(k)} A_z Q_z^{(k)^{\mathsf{T}}}$, for some $Q_z^{(k)}$ chosen uniformly at random from $\mathcal{P}$, effectively shuffling the vertex labels of the connectome. Then, we try to graph match A_z to $B_z^{(k)}$, for $z \in \{e, c\}$ and for $k = 1, 2, \ldots, 1000$, that is, we repeat the experiment 1000 times. We define accuracy as the fraction of vertices correctly assigned. We always start with the doubly flat matrix.

Figure 5 displays the results of `FAQ` along with three previous state-of-the-art algorithms on the two connectomes: (i) Umeyama's algorithm`U`, (ii) a quadratic convex relaxation `QCV` (which follows from relaxing the permutation matrix constraint to the doubly stochastic constraint in Eq. (1c)), and (iii) `PATH`. The top left panel indicates that `FAQ` *always* found the optimal solution for the chemical connectome, whereas none of the other algorithms *ever* found the optimal solution. On the other hand, the top right panel shows that for the electrical connectome, none of the four algorithms ever found the optimal. One hundred restarts of `FAQ` failed to significantly improve the results.

The bottom panels compare the wall time of the various algorithms, running on an 2.2 GHz Apple MacBook. Note that we have only a Matlab implementation of `FAQ`, whereas the other algorithms are implemented in C. Nonetheless, `FAQ` runs nearly as quickly as both `U` and `QCV`, and significantly faster than `PATH`, for both connectomes.

The properties of these connectomes are analyzed in [35]; a cursory evaluation of the properties of these graphs does not suggest to us why the chemical connectome was so much easier to graph match than the electrical one.

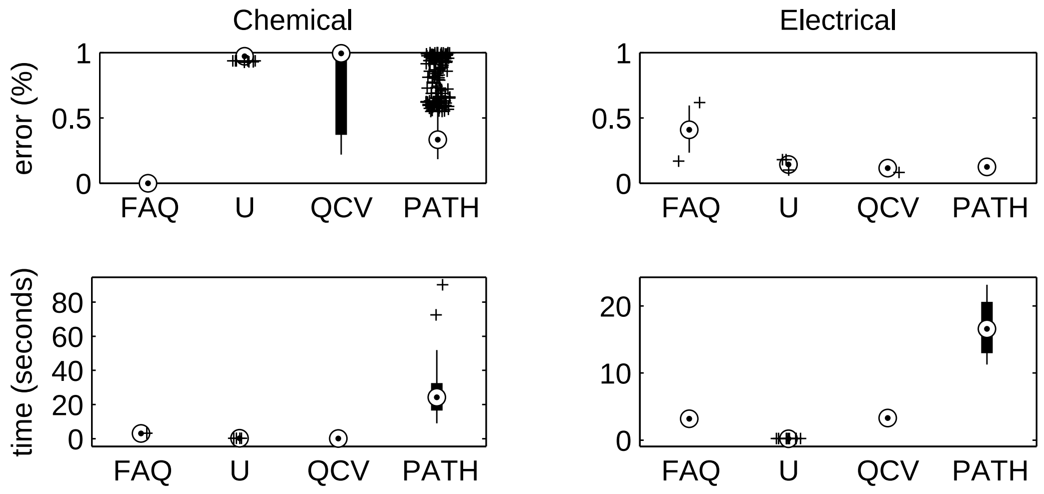

Figure 5: Performance of `U`, `QCV`, `PATH`, and `FAQ` on synthetic C. elegans connectome data, that is, graph matching the true connectomes with permuted versions of themselves. Error is the fraction of vertices correctly matched. Circle indicates the median, thick black bars indicate the quartiles, thin black lines indicate extreme but non-outlier points, and plus signs are outliers. The top panels indicate error (fraction of misassigned vertices), and the bottom panels indicate wall time on a 2.2 GHz Apple MacBook. The left panels show the chemical connectome results, and the right panels show the electrical connectome results. For the chemical connectome, `FAQ` always obtained the optimal solution, whereas none of the other algorithms ever found the optimal. On the other hand, for the electrical connectome, none of the algorithms ever found the optimal. `FAQ` also ran very quickly, nearly as quickly as `U` and `QCV`, and much faster than `PATH`, even though our `FAQ` implementation is in Matlab, and the others are in C.

To investigate the performance of FAQ on undirected graphs, we ran FAQ on binarized symmeterized versions of the graphs ($A_{ij;z} = 1$ if and only if $A_{ij;z} \geq 1$ or $A_{ji;z} \geq 1$). The resulting errors are nearly identical to those presented in Figure 5, although speed increased by greater than a factor of two. Note that the number of vertices in this brain-graph matching problem—279—is several times larger than the largest of the 32 benchmarks used above.

6 Discussion

This work presents a fast approximate quadratic assignment problem algorithm called FAQ for approximately solving large graph matching problems, motivated by brain-graphs. Our key insight was to relax the binary constraint of QAP to its continuous and non-negative counterpart—the doubly stochastic matrix—which is the convex hull of the original feasible region. Numerically, we demonstrated that not only is FAQ cubic in time, but also its leading constants are quite small—10^{-9}—suggesting that it can be used for graphs with hundreds or thousands of vertices. Moreover, it achieves better performance than previous state-of-the-art cubic-time algorithms on 29 of the 32 standard QAP benchmarks, including both directed and undirected graph matching problems. Because rQAP is non-convex, we also consider multiple restarts, and achieve improved performance for the remaining three benchmarks using only two or three restarts. We then demonstrate that the solution to our relaxed optimization problem, rQAP, is identical to that for QAP whenever the two graphs are simple and isomorphic to one another. Finally, we used it to match C. elegans connectomes to permuted versions of themselves. For the chemical connectome, of the four state-of-the-art algorithms considered, FAQ achieved perfect performance 100% of the time, whereas none of the other three algorithms ever achieved perfect performance. On the other hand, all the algorithms struggled with the electrical connectome. Moreover, FAQ ran about as fast as two of them, and significantly faster than PATH, even though FAQ is implemented in Matlab, and the others are implemented in C. Note that these connectomes have 302 vertices, several-fold more than even the largest benchmarks.

Fortunately, our work is not done. 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 1, this algorithm would take about 20 years to finish (on a standard laptop from 2011) when $n = 100,000$. We hope to be able to approximately solve rQAP on graphs much larger than that, given that the number of neurons even in a fly brain, for example, is $\approx 250,000$. More efficient algorithms and/or implementations are required for such massive graph matching.

Additional future work might generalize FAQ in a number of ways. First, many (brain-) graphs of interest will be errorfully observed [36], that is, vertices might be missing and putative edges might exhibit both false positives and false negatives. Explicitly dealing with this error source is both theoretically and practically of interest [13]. Second, for many brain-graph matching problems, the number of vertices will not be the same across the brains. Recent work from [26, 37] and [38] suggest that extensions in this direction would be both relatively straightforward and effective. Third, 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 [39]. Fourth, our matrices have certain special properties, namely sparsity, which makes more efficient algorithms (such as "active set" algorithms) readily available for further speed increases. Fourth, 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. Finally, although this approach natively operates on both unweighted and weighted graphs, multi-graphs are a possible extension.

In conclusion, this manuscript has presented an algorithm for approximately solving the quadratic assignment problem that is fast, effective, and easily generalizable. Yet, the $\mathcal{O}(n^3)$ complexity remains too slow to solve many problems of interest. 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.

A Linear Assignment Problems

The standard way of writing a Linear Assignment Problem (LAP) is

$$\text{(LAP)} \quad \underset{\pi}{\text{minimize}} \sum_{u,v\in[n]} a_{u\pi(v)} b_{ij} \tag{10a}$$

$$\text{subject to } \pi \in \Pi, \tag{10b}$$

which can be written equivalently in a number of ways using the notion of permutation matrix introduced in the main text:

$$\underset{P\in\mathcal{P}}{\text{argmin}} \, \|PA - B\|_F = \tag{11a}$$

$$\underset{P\in\mathcal{P}}{\text{argmin}} \; tr(PA - B)^{\mathsf{T}}(PA - B) = \tag{11b}$$

$$\underset{P\in\mathcal{P}}{\text{argmin}} -tr(PAB^{\mathsf{T}}) = \underset{P\in\mathcal{P}}{\text{argmin}} -\langle P^{\mathsf{T}}, AB^{\mathsf{T}}\rangle = \tag{11c}$$

$$\underset{P\in\mathcal{P}}{\text{argmin}} -\langle P, AB^{\mathsf{T}}\rangle, \tag{11d}$$

where $\langle\cdot,\cdot\rangle$ is the usual Euclidean inner product, i.e., $\langle X, Y\rangle \overset{\triangle}{=} tr(X^{\mathsf{T}}Y) = \sum_{ij} x_{ij}y_{ij}$. While the objective function and the first two constraints of LAP are linear, the binary constraints make solving even this problem computationally tricky. Nonetheless, in the last several decades, there has been much progress in accelerating algorithms for solving LAPs, starting with exponential time, all the way down to $\mathcal{O}(n^3)$ for general LAPs, and even faster for certain special cases (e.g., sparse matrices) [25, 3].

That Eq. (6b) is a LAP is evident by considering Eq. (11d). If $A = \nabla_P^{(i)}$ and $B = I$ (the $n \times n$ identity matrix), then Eq. (6b) is identical to Eq. (11d).

To solve a LAP, consider a continuous relaxation of LAP, specifically, relaxing the permutation matrix constraint to a doubly stochastic matrix constraint:

$$\text{(rLAP)} \quad \underset{P}{\text{minimize}} \quad -\langle P, AB^{\mathsf{T}}\rangle \tag{12a}$$

$$\text{subject to} \quad P \in \mathcal{D}. \tag{12b}$$

As it turns out, solving rLAP is equivalent to solving LAP.

Proposition 1. *LAP and rLAP are equivalent, meaning that they have the same optimal objective function value.*

Proof. Although this proposition is typically proven by invoking total unimodularity, we present a simpler proof here. Let P' be a solution to LAP and let $P = \sum_{i\in[k]} \alpha_i P^{(i)}$ be a solution to rLAP for some positive integer k, permutation matrices $\{P^{(i)}\}_{i\in[k]}$, and positive real numbers $\{\alpha_i\}_{i\in[k]}$ such that $\sum_{i\in[k]} \alpha_i = 1$. Note that

$$\begin{aligned}\langle P, AB^{\mathsf{T}}\rangle = \langle \sum_{i\in[k]} \alpha_i P^{(i)}, AB^{\mathsf{T}}\rangle &= \sum_{i\in[k]} \alpha_i \langle P^{(i)}, AB^{\mathsf{T}}\rangle \\ &\le \sum_{i\in[k]} \alpha_i \langle P', AB^{\mathsf{T}}\rangle = \langle P', AB^{\mathsf{T}}\rangle \le \langle P, AB^{\mathsf{T}}\rangle,\end{aligned}$$

because P' is feasible in rLAP. □

This relaxation motivates our approach to approximating QAP.

Acknowledgment

The authors would like to acknowledge two helpful reviewers as well as Lav Varshney for providing the data. This work was partially supported by the Research Program in Applied Neuroscience.

References

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

[2] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998. [Online]. Available: http://books.google.com/books?hl=en&amp;lr=&amp;id=u1RmDoJqkF4C&amp;oi=fnd&amp;pg=PR11&amp;dq=Combinatorial+optimization:+algorithms+and+complexity&amp;ots=S8xMKZ2Xyb&amp;sig=_bloCXwzNDmQS7XEt3oYM9zVAHc

[3] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. SIAM, 2009. [Online]. Available: http://books.google.com/books?id=nHIzbApLOr0C&pgis=1

[4] 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

[5] O. Sporns, G. Tononi, and R. Kotter, "The Human Connectome: A Structural Description of the Human Brain," *PLoS Computational Biology*, vol. 1, no. 4, p. e42, 2005.

[6] P. Hagmann, "From diffusion MRI to brain connectomics," Ph.D. dissertation, Institut de traitement des signaux, 2005.

[7] X.-n. N. Zuo, R. Ehmke, M. Mennes, D. Imperati, F. X. Castellanos, O. Sporns, and M. P. Milham, "Network Centrality in the Human Functional Connectome." *Cerebral cortex (New York, N.Y. : 1991)*, pp. bhr269–, Oct. 2011. [Online]. Available: http://cercor.oxfordjournals.org/cgi/content/abstract/bhr269v2

[8] J. G. Csernansky, L. Wang, S. C. Joshi, J. T. Ratnanather, and M. I. Miller, "Computational anatomy and neuropsychiatric disease: probabilistic assessment of variation and statistical inference of group difference, hemispheric asymmetry, and time-dependent change." *NeuroImage*, vol. 23 Suppl 1, pp. S56–68, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.neuroimage.2004.07.025

[9] M. Kubicki, R. McCarley, C.-F. Westin, H.-J. Park, S. Maier, R. Kikinis, F. A. Jolesz, and M. E. Shenton, "A review of diffusion tensor imaging studies in schizophrenia." *Journal of Psychiatric Research*, vol. 41, no. 1-2, pp. 15–30, 2007. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2768134&tool=pmcentrez&rendertype=abstract

[10] V. D. Calhoun, J. Sui, K. Kiehl, J. Turner, E. a. Allen, and G. Pearlson, "Exploring the psychosis functional connectome: aberrant intrinsic networks in schizophrenia and bipolar disorder." *Frontiers in psychiatry / Frontiers Research Foundation*, vol. 2, p. 75, Jan. 2011. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3254121&tool=pmcentrez&rendertype=abstract

[11] A. Fornito and E. T. Bullmore, “Connectomic intermediate phenotypes for psychiatric disorders.” *Frontiers in psychiatry / Frontiers Research Foundation*, vol. 3, p. 32, Jan. 2012. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3329878&tool=pmcentrez&rendertype=abstract

[12] A. Fornito, A. Zalesky, C. Pantelis, and E. T. Bullmore, “Schizophrenia, neuroimaging and connectomics,” *NeuroImage*, Feb. 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22387165

[13] J. T. Vogelstein and C. E. Priebe, “Shuffled Graph Classification: Theory and Connectome Applications,” *Submitted to IEEE PAMI*, 2011.

[14] R. P. W. Duin, E. Pkalska, and E. Pkalskab, “The dissimilarity space: Bridging structural and statistical pattern recognition,” *Pattern Recognition Letters*, vol. in press, no. April, May 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167865511001322http://www.sciencedirect.com/science/article/pii/S0167865511001322http://dx.doi.org/10.1016/j.patrec.2011.04.019

[15] 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

[16] L. Babali, P. Erds, and S. M. Selkow, “RANDOM GRAPH ISOMORPHISM,” *SIAM Journal on Computing*, vol. 9, no. August, pp. 628–635, 1980.

[17] L. Babali, “Moderately Exponential Bound for Graph Isomorphism,” *Fundamentals of Computation Theory*, pp. 34–50, Aug. 1981. [Online]. Available: http://portal.acm.org/citation.cfm?id=647890.739270

[18] 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

[19] 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

[20] 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

[21] K. M. Anstreicher, “Recent advances in the solution of quadratic assignment,” *SIAM Journal on Optimization*, vol. 97, no. 1-2, pp. 27–42, 2003.

[22] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955. [Online]. Available: http://doi.wiley.com/10.1002/nav.3800020109

[23] D. Knig, “Gráfok és Mátrixok,” *Matematikai és Fizikai Lapok*, vol. 38, pp. 116–119, 1931.

[24] J. Egeváry, “Matrixok kombinatorius tulajdonságairól,” *Matematikai és Fizikai Lapok*, no. 38, pp. 16–28, 1931.

[25] R. Jonker and A. Volgenant, “A shortest augmenting path algorithm for dense and sparse linear assignment problems,” *Computing*, vol. 38, no. 4, pp. 325–340, 1987. [Online]. Available: http://www.springerlink.com/index/7003M6N54004M004.pdf

[26] M. Zaslavskiy, F. R. Bach, and J.-p. 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/

[27] 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.

[28] K. M. Anstreicher and N. W. Brixius, "A new bound for the quadratic assignment problem based on convex quadratic programming," *Mathematical Programming*, vol. 89, no. 3, pp. 341–357, Feb. 2001. [Online]. Available: http://www.springerlink.com/content/uvavn88ke7818djg/

[29] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377–388, Apr. 1996. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=491619

[30] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *Analysis*, vol. I, no. 5, 1988.

[31] Z.-Y. Liu, H. Qiao, and L. Xu, "An Extended Path Following Algorithm for Graph Matching Problem." *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1451–1456, Jan. 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22331851

[32] K. M. Anstreicher, "Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming," *Journal of Global Optimization*, pp. 471–484, 2009.

[33] 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

[34] 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.

[35] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, D. B. Chklovskii, C. Spring, and J. Farm, "Structural Properties of the Caenorhabditis elegans Neuronal Network," *PLoS Computational Biology*, vol. 7, no. 2, pp. 1–41, Feb. 2011.

[36] C. E. Priebe, J. T. Vogelstein, and D. D. Bock, "Optimizing the quantity/quality trade-off in connectome inference," *Communications in Statistics Theory and Methods*, p. 7, 2011. [Online]. Available: http://arxiv.org/abs/1108.6271

[37] M. Zaslavskiy, F. R. Bach, and J.-p. Vert, "Many-to-Many Graph Matching: A Continuous Relaxation Approach," *Machine Learning and Knowledge Discovery in Databases*, vol. 6323, pp. 515–530, 2010.

[38] F. Escolano, E. Hancock, and M. Lozano, "Graph Matching through Entropic Manifold Alignment," *Computer Vision and Pattern Recognition*, 2011.

[39] 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