Fast Approximate Quadratic Programming for Large (Brain) Graph Matching

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

J.T. Vogelstein, E.T. Harley, D.E. Fishkind, and C.E. Priebe are with the Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218.
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.
* *corresponding author; current address: Department of Statistical Science, Duke University, Durham, NC 27708; current phone number: +1-443-858-9911; current email address:* jovo@stat.duke.edu

Abstract

Quadratic assignment problems (QAPs) arise in a wide variety of domains, ranging from operations research to graph theory to computer vision to neuroscience. In the age of big data, graph valued data is becoming more prominent, and with it, a desire to run algorithms on ever larger graphs. Because QAP is **NP**-hard, exact algorithms are intractable. Approximate algorithms necessarily employ an accuracy/efficiency trade-off. We developed a fast approximate quadratic assignment algorithm (`FAQ`). `FAQ` finds a local optima in (worst case) time cubic in the number of vertices, similar to other approximate QAP algorithms. We demonstrate empirically that our algorithm is faster and achieves a lower objective value on over 80% of the suite of QAP benchmarks, compared with the previous state-of-the-art. Applying the algorithms to our motivating example, matching C. elegans connectomes (brain-graphs), we find that `FAQ` achieves the optimal performance in record time, whereas none of the others even find the optimum.

Keywords:
graph theory, neuroscience, nonlinear optimization

1. Introduction

Quadratic assignment problems (QAPs) were first devised by Koopmans and Beckmann in 1957 to solve a ubiquitous problem in distributed resource allocation [1]. Already at that time, it was recognized to be a general kind of problem with many important and disparate applications, including the ever popular traveling salesman problem. As it turns out, certain forms of 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—can be cast as a quadratic assignment problem [2]. The beauty of realizing this relationship is that we can bring to bear all the optimization theoretic tools in our toolbox to address graph matching, which is a notoriously hard problem (**NP**-hard in particular [3]).

Because GM and QAP are so difficult, communities spanning operations research, computer vision, combinatorics, and optimization theory have developed methodologies for solving these problems [4, 5]. Moreover, with the emergence of big data, large graphs are becoming an increasingly popular data structure for storing information [6]. As the number of vertices increases, we are forced to use approximate algorithms to find good (but not optimal) solutions. For any such problems, we face a necessary accuracy/efficiency trade-off: slower algorithms could achieve better performance given more time. Thus for any applied problem, given the no free lunch theorem, we search for algorithms with the appropriate trade-off. If an algorithm outperforms another on both accuracy and efficiency on the class of problems of interest, then it is clearly preferential.

We are motivated by "connectomics", an emerging discipline within neuroscience devoted to the study of brain-graphs, where vertices represent (collections of) neurons and edges represent connections between them [7, 8]. Typically, in human connectomics, the brain is subdivided into approximately 100 vertices (regions), even though it consists of approximately 86 billions vertices (neurons) [9]. At the other end of the spectrum, the small hermaphroditic *Caenorhabditis elegans* (*C. elegans*) has only 302 neurons. Thus, regardless of which brains are under investigation, they are currently typically represented by graphs with $\mathcal{O}(100)$ vertices.

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 [10]. 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 [11, 12, 13, 14]. Part of the reason for the lack of publications comparing brain-graphs is because algorithms for matching graphs of this size were ineffective. Thus, currently available tests for connectopic explanation of psychiatric disorders hedge upon first choosing some number of graph invariants to compare across populations, rather than comparing the graphs directly. The graph invariant approach to classifying is both theoretically and practically inferior to comparing whole graphs via matching [15].

More generally, state-of-the-art inference procedures for essentially any decision-theoretic or inference task follow from constructing interpoint dissimilarity matrices, or graphs [16]. 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 [4].

The remainder of this paper is organized as follows. Section 2.1 formally defines the QAP and a relaxation thereof that we will operate under. Section 2.3 defines graph matching, and explains how it can be solved via QAP. Section 3 describes our algorithm, a Fast Approximate QAP (`FAQ`). Section 4 provides a number of theoretical and empirical results, comparing our algorithm to previous state-of-the-art algorithms in terms of both computational efficiency and objective function value for various QAPs. This section concludes with an analysis of `FAQ` on our motivating problem. We conclude with a discussion in Section 5.

2. Preliminaries

2.1. Quadratic Assignment Problems

A quadratic assignment problem can be stated thusly. Let $A = (a_{uv})$ and $B = (b_{uv})$ both be $n \times n$ matrices. Moreover, let Π be the set of permutation functions (bijections), $\Pi = \{\pi : [n] \to [n]\}$, where $[n] = \{1, \ldots, n\}$. We therefore can write the Koopmans-Beckmann version of QAP:

$$\text{(KB)} \qquad \begin{array}{ll} \text{minimize} & \sum_{u,v \in [n]} b_{uv} a_{\pi(u)\pi(v)} \\ \text{subject to} & \pi \in \Pi \end{array} \tag{1}$$

(sometimes an additional linear cost function is added, but we drop it here for brevity).

Equation 1 can be rewritten in matrix notation given the following definitions. 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. Thus, we can write $PAP^\mathsf{T} = (a_{\pi(u)\pi(v)})$ whenever P is the permutation matrix corresponding to the bijection π, yielding the following equivalent optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{u,v \in [n]} b_{uv}(p_{vu} a_{uv} p_{uv}) \\ \text{subject to} & P \in \mathcal{P}. \end{array}$$

Moreover, the sum of entry-wise products of elements can be written as the trace. Therefore, we can rewrite the QAP in the trace formulation, which we hereafter refer to as the QAP optimization function:

$$\text{(QAP)} \qquad \begin{array}{ll} \text{minimize} & -tr(B^\mathsf{T} P^\mathsf{T} A P) \\ \text{subject to} & P \in \mathcal{P}. \end{array} \tag{2}$$

2.2. Relaxed Quadratic Assignment Problem

Eq. 2 is a quadratic problem with linear and binary constraints (note that it can also be written as a quadratic problem with quadratic constraints, because $p_{uv} \in \{0,1\}$ is equivalent to $p_{uv} = p_{uv}^2$). Thus, one could use any number of algorithms for solving quadratic problems with binary (or quadratic) constraints. Because of the constraints on the feasible region, finding a *global* optimum is **NP**-hard. Instead, one could search for a *local* optimum, for example, using a projected gradient method, which descends along the gradient projected into the feasible region. While feasible, projecting into the set of permutation matrices may move the current iterate's estimate to somewhere that increases the objective function value. Moreover, it is not clear how to directly find a permutation matrix that lowers the objective function value. We therefore adopt a different strategy.

Rather than directly searching for a permutation matrix, we relax the constraint set to the convex hull of the set of permutation matrices, the Birkhoff polytope. The Birkhoff polytope is the set of 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. We therefore obtain the following relaxed quadratic assignment problem:

$$\text{(rQAP)} \qquad \begin{array}{ll} \text{minimize} & -tr(B^\mathsf{T} P^\mathsf{T} A P) \\ \text{subject to} & P \in \mathcal{D}. \end{array} \tag{3}$$

Although rQAP is a quadratic problem with *linear* constraints, it is not necessarily convex. The objective function, $f(P) = -tr(B^\mathsf{T} P^\mathsf{T} A P)$, has a Hessian that is not necessarily positive definite:

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

where $\otimes$ indicates the Kronecker product. This means that the solution space will potentially be multimodal, making initialization important.

2.3. 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 $|\mathcal{V}_A| = |\mathcal{V}_B| = n$, 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 **P** [17]. In fact, GI is one of the few problems for which, if **P**≠**NP**, then GI might reside in an intermediate complexity class called **GI**-complete. GM, however, is known to be **NP**-hard. Yet, for large classes of GI and GM problems, linear or polynomial time algorithms are available [18]. Moreover, at worst, it is clear that GI is only "moderately exponential," for example, $\mathcal{O}(\exp\{n^{1/2+o(1)}\})$ [19]. Unfortunately, even when linear or polynomial time GI or GM algorithms are available for special cases of graphs, the constants are often 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!) [20].

Because we are interested in solving GM for graphs with $\dot{\approx}10^6$ 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.

2.4. Graph Matching as a Quadratic Assignment Problem

We can formally write the graph matching problem as an optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{u,v\in[n]}(a_{uv} - b_{\pi(u)\pi(v)})^2 \\ \text{subject to} & \pi \in \Pi. \end{array}$$

Using the same linear algebra notation as above, we can rewrite the objective function,

$$\begin{aligned}(a_{uv} - b_{\pi(u)\pi(v)})^2 = \left\|A - PBP^{\mathsf{T}}\right\|_F^2 \\ = tr\{(A - PBP^{\mathsf{T}})^{\mathsf{T}}(A - PBP^{\mathsf{T}})\}. \quad (4)\end{aligned}$$

Dropping irrelevant constants, we obtain the trace formulation of the graph matching problem:

$$\text{(GM)} \qquad \begin{array}{ll} \text{minimize} & -tr(B^\mathsf{T} P^\mathsf{T} A P) \\ \text{subject to} & P \in \mathcal{P}. \end{array} \tag{5}$$

Clearly, the objective function for GM is just the negative of the objective function for QAP. Thus, any descent algorithm for the former can be directly applied to the latter. Moreover, any approximate algorithms also trivially apply to analogous approximations.

3. 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 [21, 22].

Although `FW` is a relatively standard solver, especially as a subroutine for QAP algorithms [23], 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

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

Step 2: Compute the new direction $Q^{(i)}$: The new direction 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

$$\widetilde{f}^{(i)}(P) \stackrel{\triangle}{=} f(P^{(i)}) + \nabla f(P^{(i)})^\mathsf{T}(P - P^{(i)}). \tag{6}$$

Dropping terms independent of P, we obtain the following sub-problem:

$$\begin{array}{ll} \text{minimize} & \nabla f(P^{(i)})^\intercal P \\ \text{subject to} & P \in \mathcal{D}. \end{array} \tag{7}$$

Let $Q^{(i)}$ indicate the argmin of Eq. (7). As it turns out, Eq. (7) can be solved as a *Linear Assignment Problem* (LAP). The details of LAPs are well known [5], 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 [24, 25, 26]. 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 [27, 5]. 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 $Q^{(i)}$, the new point is given maximizing the *original* optimization problem, rQAP, along the line segment from $P^{(i)}$ to $Q^{(i)}$ in $\mathcal{D}$.

$$\begin{array}{ll} \text{minimize} & f(P^{(i)} + \alpha^{(i)} Q^{(i)}) \\ \text{subject to} & \alpha \in [0, 1]. \end{array} \tag{8}$$

Let $\alpha^{(i)}$ indicate the argmin of Eq. (8). 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

$$P^{(i+1)} = P^{(i)} + \alpha^{(i)} Q^{(i)}. \tag{9}$$

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 $P^{(i_{max})}$ be the doubly stochastic matrix resulting from the final iteration of `FW`. We project $P^{(i_{max})}$ onto the set of permutation matrices, yielding

$$\begin{array}{ll} \text{minimize} & -\langle P^{(i_{max})}, P \rangle \\ \text{subject to} & P \in \mathcal{P}. \end{array} \tag{10}$$

where $\langle \cdot, \cdot \rangle$ is the usual Euclidean inner product, i.e., $\langle X, Y \rangle \stackrel{\triangle}{=} tr(X^\intercal Y) = \sum_{ij} x_{ij} y_{ij}$. Note that Eq. (10) is a LAP (again, see appendix for details).

Pseudocode 1 provides pseudocode for the whole algorithm.

Pseudocode 1 `FAQ` for finding a local optimum of rQAP

Input: graphs A and B as well as stopping criteria
Output: $\widehat{P}$, an estimated permutation matrix

1: Choose an initialization, $P^{(0)} = \mathbf{1}\mathbf{1}^\intercal/n$
2: **while** stopping criteria not met **do**
3: Compute the gradient of f at the current point via Eq. (6)
4: Compute the direction $Q^{(i)}$ by solving Eq. (7) via the Hungarian algorithm
5: Compute the step size $\alpha^{(i)}$ by solving Eq. (8)
6: Update $P^{(i)}$ according to Eq. (9)
7: **end while**
8: Obtain $\widehat{P}$ by solving Eq. (10) via the Hungarian algorithm.

4. Results

4.1. 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, at least for a particular simulation setting. The primary computational bottleneck of `FAQ` is solving the LAP as a subroutine. We use the Jonker and Volgenant version of the Hungarian algorithm [27, 5], which is known to scale cubically in the number of vertices or better. Figure 1 suggests that `FAQ` is not just cubic in time, but also has very small leading constants ($\dot{\approx}10^{-9}$ seconds), making using this algorithm feasible for even reasonably large graphs. Note that the other state-of-the-art approximate graph matching algorithms also have cubic or worse time complexity in the number of vertices. We will describe these other algorithms and their time complexity in greater detail below.

4.2. QAP Benchmark Accuracy

Having demonstrated both theoretically and empirically the `FAQ` has cubic time complexity, we next decided to evaluate its accuracy on a suite of standard benchmarks. More specifically, QAPLIB is a library of 137 quadratic assignment problems, ranging in size from 10 to 256 vertices [28]. Recent graph matching papers typically evaluate the performance of their algorithm on 16 of the benchmarks that are known to be "particularly difficult" [29, 30]. We compare the results of `FAQ` to the results of four other

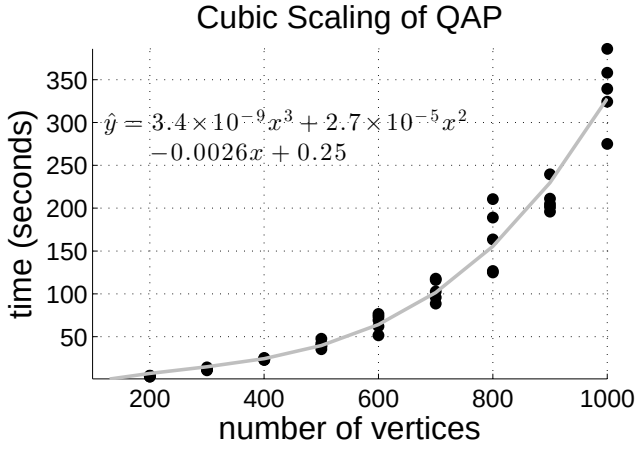

Figure 1: Running time of FAQ as function of number of vertices. Data was sampled from an Erdös-Rényi model with $p = log(n)/n$. Each dot represents a single simulation, with 100 simulations per n. The solid line is the best fit cubic function. Note the leading constant is $\doteq 10^{-9}$ seconds. FAQ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

state-of-the-art graph matching algorithms: (1) the PATH algorithm, which solves a path between a convex and concave relaxation of QAP [29], (2) QCV which is the convex relaxation used to initialize the PATH algorithm, (3) the RANK algorithm [31], which uses a spectral decomposition, and (4) the Umeyama algorithm (denoted by U henceforth), which also uses a spectral decomposition [2]. We chose these four algorithms to compare because the code is freely available from the `graphm` package [29]. Figure 2 plots the logarithm (base 10, here and elsewhere) of the relative accuracy, that is, $\log_{10}(\hat{f}_{FAQ}/\hat{f}_X)$, for $X \in \{$`PATH, QCV, RANK, U, all`$\}$, where `all` is just the best performer of all the non FAQ algorithms. Clearly, FAQ does significantly better than all the other algorithms, outperforming all of them on $\approx 94\%$ of the problems, often by nearly an order of magnitude in terms of relative error.

4.3. QAP Benchmark Efficiency

As we mentioned in the introduction, the quality of an *approximate* algorithm depends not just on its accuracy, but also its efficiency. Therefore, we compare the wall time of each of the five algorithms on all 137 benchmarks in Figure 3. We fit an iteratively weighted least squares linear regression function (Matlab's `robustfit`) to regress the logarithm of time (in seconds) onto the logarithm of the number of vertices. The numbers beside the lines

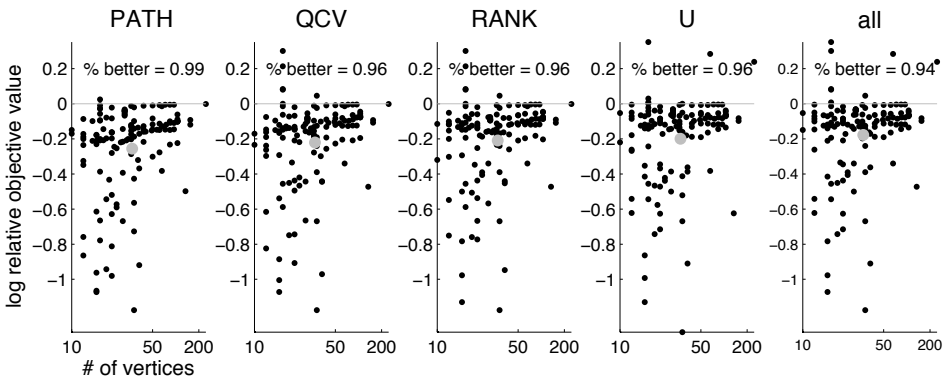

Figure 2: Relative accuracy—defined to be $\log_{10}(\hat{f}_{FAQ}/\hat{f}_X)$—of all the four algorithms compared with FAQ. Note that FAQ is better than all the other algorithms on $\approx 94\%$ of the benchmarks. The abscissa is the log number of vertices. The gray dot indicates the mean improvement of FAQ over the other algorithms.

indicate the slopes of the regression functions. The PATH algorithm has the worst slope. QCV and FAQ have nearly identical slopes, which makes sense, given that the are solving very similar objective functions. Similarly, RANK and U have very similar slopes; they are both using spectral approaches. Note, however, that although the slope of RANK and U are smaller than that of FAQ, they both seem to be super linear on this log-log plot, suggesting that as the number of vertices increases, their compute time might exceed that of the other algorithms. Regardless, when comparing approximation algorithms, it is the speed/accuracy trade-off that is most important, in particular, in the problem space of interest. For graphs with hundreds of vertices, all of these algorithms are sufficiently fast to use, so performance in terms of speed will be the deciding factor for many applications. Of note is that the FAQ algorithm has a relatively high variance for these problems. This is due to the number of Hungarian algorithms performed, which is determined by the "difficulty" of the problem to converge. We could fix the number of Hungarian algorithms, in which case the variance would decrease dramatically. For our application, this variance is not problematic.

4.4. QAP Benchmark Accuracy/Efficiency Trade-off

In the PATH, the authors demonstrated that PATH outperformed QCV and U on a variety of simulated and real examples in terms of objective function [29]. If FAQ yields a lower objective function value than FAQ, and is faster, then it clearly is superior to PATH on such problems. Figure 4 compares

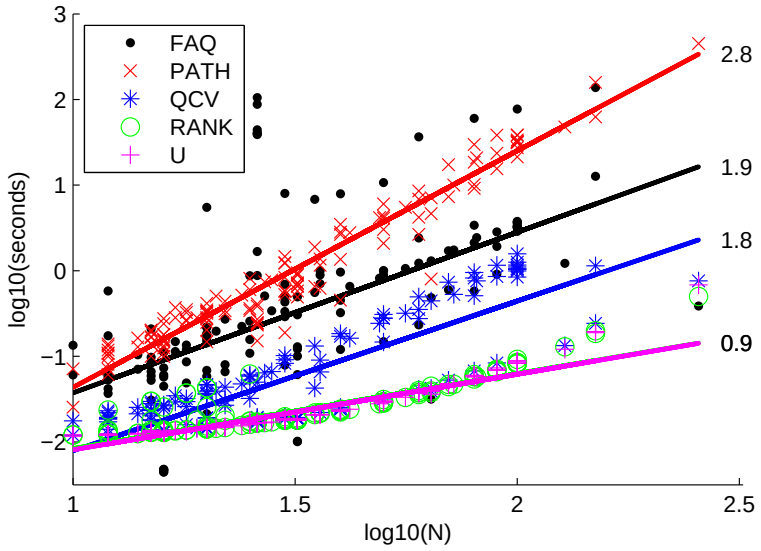

Figure 3: Absolute wall time for running each of the five algorithms on all 137 benchmarks. We fit a line on this log-log plot for each algorithm; the slope is displayed beside each line. The FAQ slope is much better than the PATH slope, and worse than the others. Note, however, the time for RANK and U appears to be superlinear on this log-log plot, suggesting that perhaps as the number of vertices increases, PATH might be faster.

the performance of FAQ with PATH along both dimensions of performance—accuracy and efficiency—for all 137 benchmarks in the QAPLIB library. The right panel indicates that FAQ is both more accurate and more efficient on 80% of the problems.

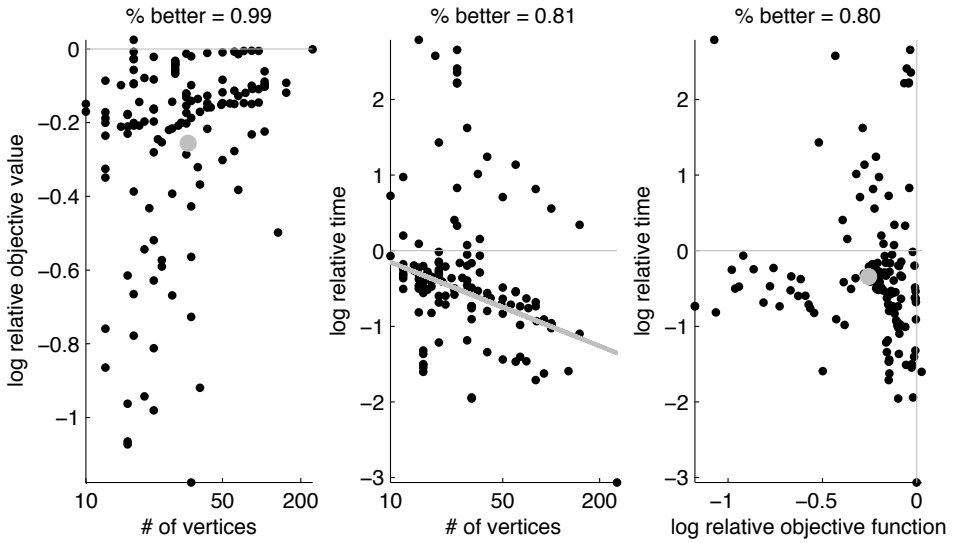

Figure 4: Comparison of FAQ with PATH in terms of both accuracy and efficiency. The left panel is the same as the left panel of Figure 2. The middle plots the relative wall time of FAQ to PATH as a function of the number of vertices, also on a log-log scale. The gray line is the best fit slope on this plot, suggesting that FAQ is getting exponentially faster than PATH as the number of vertices gets larger. Finally, the right panel plots log relative time versus log relative objective function value, demonstrating that FAQ outperforms PATH on both dimensions on 80% of the benchmarks.

4.5. QAP Directed Benchmarks

Recently, Liu et al. [32] proposed a modification of the PATH algorithm that adjusted PATH to be more appropriate for directed graphs, as the theory motivating the development of PATH relied upon the graphs being undirected. FAQ, on the other hand, does not depend on the graphs being simple; rather, directed or weighted graphs are both unproblematic. Liu et al. compare the performance of their algorithm (EPATH) with U, QCV, and GRAD on the set of 16 particularly difficult directed benchmarks from QAPLIB. 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 1 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.

Table 1: Comparison of FAQ with optimal objective function value and previous state-of-the-art for directed 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***

4.6. Theoretical properties of FAQ

In addition to guarantees on computational time, we have a guarantee on performance:

Lemma 1. *If A and B are the adjacency matrices of simple graphs (symmetric, hollow, and binary) that are isomorphic to one another, then the minimum of rQAP is equal to the minimum of QAP.*

Proof. Because any feasible solution to GM 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 $A = PBP^\intercal$, so that $\langle A, PBP^\intercal \rangle = 2m$, 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 $(DBD^\mathsf{T})_{uv} \geq 1$ 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. □

4.7. Multiple Restarts

Although `FAQ` outperformed all other algorithms on nearly every benchmark, that `FAQ` was not always the best was annoying to us. We therefore utilized 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). We can utilize the non-convexity as a feature, however, whenever (i) we have some reason to believe that better solutions exist (many algorithms efficiently compute relatively tight lower bounds [33]), 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 doubly flat matrix, and satisfy the conditions. Therefore, we sample $K \in \mathcal{D}$, a random doubly stochastic matrix using 10 iterations of Sinkhorn balancing [34], 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. Fixing the number of restarts, we still have a cubic time algorithm, although the constants change.

Table 2 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 of the five algorithms (running only `FAQ` once). Note that we only consider the 16 particularly difficult benchmarks for this evaluation. `FAQ` only required three restarts to outperform all other approximate algorithms on all 16 of 16 difficult benchmarks. Moreover, after 100 restarts, `FAQ` finds the absolute minimum on 3 of the 16 benchmarks; none of the other algorithms ever achieved the absolute minimum on any of these benchmarks.

4.8. Brain-Graph Matching

A "chemical connectome" is a brain-graph in which vertices correspond to (collections of) neurons, and edges correspond to chemical synapses between them. The *Caenorhabditis elegans* (*C. elegans*) is a small worm (nematode) with 302 labeled vertices (in the hermaphroditic sex). We consider

Table 2: Comparison of FAQ with optimal objective function value and the best result on the 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 previous state-of-the-art on all 16 particularly difficult benchmarks.

#	Problem	Optimal	FAQ_{100}	FAQ_{3}	previous min
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

the subgraph with 279 somatic neurons that form edges with other neurons [35, 36].

Because pairs of neurons sometimes have multiple synapses between them, and they are directed, the chemical connectome of *C. elegans* may be thought of as a weighted directed graph. We therefore conducted the following synthetic experiments. Let $A_{uv} \in \{0, 1, 2, \ldots\}$ be the number of synapses from neuron u to neuron v, and let $A = \{A_{uv}\}_{u,v\in[279]}$. To generate synthetic data, we let $B^{(k)} = Q^{(k)} A {Q^{(k)}}^\mathsf{T}$, for some $Q^{(k)}$ chosen uniformly at random from $\mathcal{P}$, effectively shuffling the vertex labels of the connectome. Then, we try to graph match A to $B^{(k)}$, 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 U, QCV, and PATH. The left panel indicates that FAQ *always* found the optimal solution for the chemical connectome, whereas none of the other algorithms *ever* found the optimal

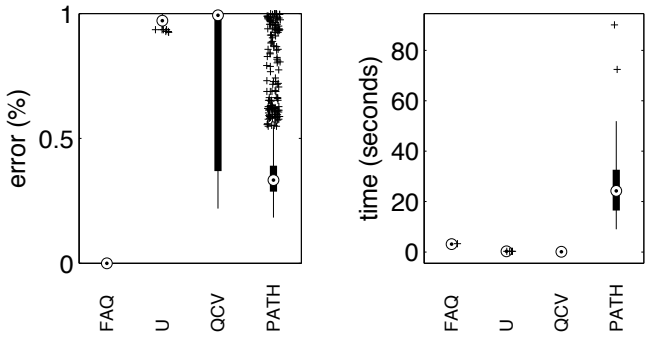

Figure 5: Performance of U, Qcv, Path, and Faq on synthetic C. elegans connectome data, that is, graph matching the true connectomes with permuted versions of themselves. Error is the fraction of vertices correctly matched. Circle indicates the median, thick black bars indicate the quartiles, thin black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

solution. The right panel compares 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. Unlike in the QAPLIB benchmarks, Faq runs nearly as quickly as both U and Qcv; and as expected, Faq runs significantly faster than Path.

To investigate the performance of Faq on undirected graphs, we ran Faq on binarized symmeterized versions of the graphs ($A_{uv} = 1$ if and only if $A_{uv} \geq 1$ or $A_{vu} \geq 1$). 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 larger than the largest of the 137 benchmarks used above.

5. Discussion

5.1. Summary

This work presents a fast approximate quadratic assignment problem algorithm called Faq for approximately solving large quadratic assignment problems, motivated by brain-graph matching. 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. 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 (§4.1).

Moreover, it achieves better accuracy than previous state-of-the-art approximate algorithms on on over 93% of the 137 QAPLIB benchmarks (§4.2), is faster than PATH(§4.3), and is both faster and achieves at least as low performance on over 80% of the benchmarks (§4.4), including both directed and undirected graph matching problems (§4.5). In addition to the theoretical guarantees of cubic run time, we also 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 (§4.6). Because rQAP is non-convex, we also consider multiple restarts, and achieve improved performance for the particularly difficult benchmarks using only two or three restarts (§4.7).

Finally, we used it to match C. elegans connectomes to permuted versions of themselves (§4.8). 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. 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 279 vertices, more vertices than even the largest benchmarks.

5.2. Related Work

Our approach is quite similar to other approaches that have recently appeared in the literature. Perhaps its closest cousins include [29, 37] and [38], which are all of the "PATH" following variety. Zaslavskiy et al. seems to consider but discard FAQ [29] because they did not like projecting onto the set of permutations matrices. Their solution, while elegant, is both slower and obtains a worse objective function value on nearly all benchmark problems. Others have considered similar relaxations, but usually in the context of finding lower bounds [39] or as subroutines for finding exact solutions [40]. Our work seems to be the first to utilize the precise algorithm described in Pseudocode 1 to find fast approximate solutions to QAP.

5.3. Future Work

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. Although a few other state-of-the-art algorithms were more efficient than `FAQ`, their accuracy was significantly worse. So the search continues to find approximate graph matching algorithms with scaling rules like `QCV`, `U` or `RANK`, but performance like `FAQ`.

Additional future work might generalize `FAQ` in a number of ways. First, many (brain-) graphs of interest will be errorfully observed [41], that is, vertices might be missing and putative edges might exhibit both false positives and negatives. Explicitly dealing with this error source is both theoretically and practically of interest [15]. Second, for many brain-graph matching problems, the number of vertices will not be the same across the brains. Recent work from [29, 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 linear optimization problem with linear constraints. A number of parallelized optimization strategies could therefore potentially be brought to bear on this problem [42]. 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. Fifth, 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.

5.4. Concluding Thoughts

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`.

Appendix A. Linear Assignment Problems

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

$$\begin{array}{ll} \text{minimize} & \sum_{u,v\in[n]} a_{u\pi(v)} b_{uv} \\ \text{subject to} & P \in \mathcal{P}. \end{array}$$

The LAP objective function, like the QAP objective function, enjoys a number of equivalent formulations, including

$$\text{(LAP)} \qquad \begin{array}{ll} \text{minimize} & \langle P, AB^{\mathsf{T}} \rangle \\ \text{subject to} & P \in \mathcal{P}. \end{array} \tag{A.1}$$

The binary constraints of LAP—like those of QAP—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) [27, 5].

To see that Eq. (7) is identical to Eq. (A.2), simply let $A = \nabla_P^{(i)}$ and $B = I$ (the $n \times n$ identity matrix).

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

$$\text{(rLAP)} \qquad \begin{array}{ll} \text{minimize} & \langle P, AB^{\mathsf{T}} \rangle \\ \text{subject to} & P \in \mathcal{D}. \end{array} \tag{A.2}$$

As it turns out, the minima of LAP and rLAP are equal to one anther [5]. This relaxation motivates our approach to approximating QAP.

Acknowledgment

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

References

[1] T. C. T. C. Koopmans, M. Beckman, M. Beckmann, Assignment Problems and the Location of Economic Activities, The Econometric Society 25 (1957) 53–76.

[2] S. Umeyama, An Eigendecomposition Approach to Weighted Graph Matching Problems, Analysis I (1988).

[3] C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Publications, 1998.

[4] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty Years of Graph Matching in Pattern Recognition, International Journal of Pattern Recognition and Artificial Intelligence 18 (2004) 265–298.

[5] R. E. Burkard, M. Dell'Amico, S. Martello, Assignment Problems, SIAM, 2009.

[6] E. Kolaczyk, Statistical Analysis of Network Data: Methods and Models, Springer, 2010.

[7] O. Sporns, G. Tononi, R. Kotter, The Human Connectome: A Structural Description of the Human Brain, PLoS Computational Biology 1 (2005) e42.

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

[9] S. Herculano-Houzel, The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost., Proceedings of the National Academy of Sciences of the United States of America 109 Suppl (2012) 10661–8.

[10] J. G. Csernansky, L. Wang, S. C. Joshi, J. T. Ratnanather, 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 23 Suppl 1 (2004) S56–68.

[11] M. Kubicki, R. McCarley, C.-F. Westin, H.-J. Park, S. Maier, R. Kikinis, F. A. Jolesz, M. E. Shenton, A review of diffusion tensor imaging studies in schizophrenia., Journal of Psychiatric Research 41 (2007) 15–30.

[12] V. D. Calhoun, J. Sui, K. a. Kiehl, J. A. Turner, E. a. Allen, G. Pearlson, Exploring the psychosis functional connectome: aberrant intrinsic networks in schizophrenia and bipolar disorder., Frontiers in psychiatry / Frontiers Research Foundation 2 (2011) 75.

[13] A. Fornito, E. T. Bullmore, Connectomic intermediate phenotypes for psychiatric disorders., Frontiers in psychiatry / Frontiers Research Foundation 3 (2012) 32.

[14] A. Fornito, A. Zalesky, C. Pantelis, E. T. Bullmore, Schizophrenia, neuroimaging and connectomics, NeuroImage (2012).

[15] J. T. Vogelstein, C. E. Priebe, Shuffled Graph Classification: Theory and Connectome Applications, Submitted to IEEE PAMI (2011).

[16] R. P. W. Duin, E. Pkalska, E. Pkalskab, The dissimilarity space: Bridging structural and statistical pattern recognition, Pattern Recognition Letters in press (2011).

[17] S. Fortin, The Graph Isomorphism Problem, Technical Report, University of Alberta, Dept of CS (1996).

[18] L. Babali, P. Erds, S. M. Selkow, RANDOM GRAPH ISOMORPHISM, SIAM Journal on Computing 9 (1980) 628–635.

[19] L. Babali, Moderately Exponential Bound for Graph Isomorphism, Fundamentals of Computation Theory (1981) 34–50.

[20] J. Chen, A Linear-Time Algorithm for Isomorphism of Graphs of Bounded Average Genus, SIAM Journal on Discrete Mathematics 7 (1994) 614.

[21] M. Frank, P. Wolfe, An Algorithm for Quadratic Programming, Naval Research Logistics Quarterly 3 (1956) 95–110.

[22] S. P. Bradley, A. C. Hax, T. L. Magnanti, Applied Mathematical Programming, Addison-Wesley, 1977.

[23] K. M. Anstreicher, Recent advances in the solution of quadratic assignment, SIAM Journal on Optimization 97 (2003) 27–42.

[24] H. W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1955) 83–97.

[25] D. Knig, Gráfok és Mátrixok, Matematikai és Fizikai Lapok 38 (1931) 116–119.

[26] J. Egeváry, Matrixok kombinatorius tulajdonságairól, Matematikai és Fizikai Lapok 38 (1931) 16–28.

[27] R. Jonker, A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, Computing 38 (1987) 325–340.

[28] R. E. Burkard, S. E. Karisch, F. Rendl, QAPLIB A Quadratic Assignment Problem Library, Journal of Global Optimization 10 (1997) 391–403.

[29] M. Zaslavskiy, F. R. Bach, J.-p. P. Vert, A path following algorithm for the graph matching problem, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 2227–2242.

[30] C. Schellewald, S. Roth, 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.

[31] R. Singh, J. Xu, B. Berger, Pairwise global alignment of protein interaction networks by matching neighborhood topology, RESEARCH IN COMPUTATIONAL MOLECULAR BIOLOGY 4453 (2007) 16–31.

[32] Z.-Y. Liu, H. Qiao, L. Xu, An Extended Path Following Algorithm for Graph Matching Problem., IEEE transactions on pattern analysis and machine intelligence 34 (2012) 1451–1456.

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

[34] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, The Annals of Mathematical Statistics 35 (1964) 876–879.

[35] J. G. White, E. Southgate, J. N. Thomson, S. Brenner, The structure of the nervous system of the nematode caenorhabditis elegans., Philosophical Transactions of Royal Society London. Series B, Biological Sciences 314 (1986) 1–340.

[36] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, D. B. Chklovskii, C. Spring, J. Farm, Structural Properties of the Caenorhabditis elegans Neuronal Network, PLoS Computational Biology 7 (2011) 1–41.

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

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

[39] K. M. Anstreicher, N. W. Brixius, A new bound for the quadratic assignment problem based on convex quadratic programming, Mathematical Programming 89 (2001) 341–357.

[40] N. W. Brixius, Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations 1 Introduction, Management (2000) 1–20.

[41] C. E. Priebe, J. T. Vogelstein, D. D. Bock, Optimizing the quantity/quality trade-off in connectome inference, Communications in Statistics Theory and Methods (2011) 7.

[42] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Foundations and Trends in Machine Learning, Foundations and Trends in Machine Learning 3 (2011) 1–122.