

# Fast Approximate Quadratic Programming for Large (Brain) Graph Matching

Joshua T. Vogelstein<sup>1,\*</sup>, John M. Conroy<sup>2</sup>, Vince Lyzinski<sup>3</sup>, Louis J. Podrazik<sup>2</sup>, Steven G. Kratzer<sup>2</sup>, Eric T. Harley<sup>4</sup>, Donniell E. Fishkind<sup>4</sup>, R. Jacob Vogelstein<sup>5</sup> and Carey E. Priebe<sup>4</sup>

**1 Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, USA**

**2 Institute for Defense Analyses, Center for Computing Sciences, Bowie, MD, USA**

**3 Johns Hopkins University Human Language Technology Center of Excellence, Baltimore, MD, USA**

**4 Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA**

**5 Johns Hopkins University Applied Physics Laboratory, Laurel, MD, USA**

\* E-mail: jovo@jhu.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.


## 1 Introduction


In its most general form, the graph matching problem (GMP)—finding an alignment of the vertices of two graphs which minimizes the number of induced edge disagreements—is equivalent to a quadratic assignment problem (QAP) [1]. QAPs were first devised by Koopmans and Beckmann in 1957 to solve a ubiquitous problem in distributed resource allocation [2]. Since then, many important problems in combinatorial optimization (for example, the “traveling salesman problem,” and the GMP) have been shown to be specialized QAPs. Unfortunately, QAPs, in general, are known to be **NP**-hard [3]. Due to the wide applicability and computational complexity of QAPs, there is a large literature devoted to algorithmic approximations and specialized formulations of QAPs; see [4] for a comprehensive literature survey. In casting the GMP as a QAP, we bring to bear a host of optimization theoretic tools and methodologies to address the GMP: Frank-Wolfe [5], gradient-descent [6], ADMM [7], etc.

The GMP has applications in a wide variety of disciplines, spanning operations research, computer vision, combinatorics, pattern recognition, and neuroscience; see [8] for a comprehensive survey of the literature. We are motivated by applications in “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 [9, 10]. Available human connectomes have  $\mathcal{O}(10^6)$  vertices and  $\mathcal{O}(10^8)$  edges, and leverage state-of-the-art advances in DT-MRI imagery, big data processing and computer vision [11]. Contrast this with the fact that the human brain consists of approximately 86 billion vertices (neurons) [12]. At the other end of the spectrum, the small hermaphroditic *Caenorhabditis elegans* worm (*C. elegans*) has only 302 neurons, with a fully mapped connectome. Consequently, these graphs (and more broadly the emergence of the large graphs common in big data [13]) demand algorithms that both accurately match

small-to-moderately sized graphs and also scale to match very large graphs.

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 [14]. To date, however, these comparisons have largely rested on anatomical (e.g., shape) comparisons, not graph (e.g., structural) comparisons. This is despite the widely held doctrine that many psychiatric disorders are fundamentally “connectopathies,” i.e. disorders of the connections of the brain [15–18]. Part of the reason for the lack of publications structurally comparing brain-graphs is that existing algorithms for matching graphs of this size are largely 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’ structure directly. Moreover, recent results suggest that the graph invariant approach to classifying is both theoretically and practically inferior to comparing whole graphs via matching [19].

More generally, state-of-the-art inference procedures for many decision-theoretic or inference tasks follow from first constructing interpoint dissimilarity matrices, i.e. graphs [20]. 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 can be very large, we face a necessary accuracy/efficiency trade-off when approximately solving these GMPs: slower algorithms could achieve better performance given more time (at the extreme, exhaustive search algorithms have provably optimal performance). Our algorithm—Fast Approximate QAP (FAQ) algorithm for approximate GM—achieves the best available trade-off between accuracy and efficiency, outperforming the existing state-of-the-art in both accuracy and efficiency on a large proportion of QAP benchmarks and biologically inspired network matching problems. 

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 posited as a QAP. Section 3 describes the FAQ algorithm. Section 4 provides a number of theoretical and empirical results, and compares our algorithm to previous state-of-the-art algorithms. This section concludes with an analysis of FAQ on our motivating problem. We conclude with a discussion in Section 5. 

## 2 Preliminaries

In this section, we formally define the QAP and the GMP. We then show how the GMP can be recast as a special case of the QAP.

### 2.1 Quadratic Assignment Problems

We first define the general quadratic assignment problem. Let  $A = (a_{uv}), B = (b_{uv}) \in \mathbb{R}^{n \times n}$  be two  $n \times n$  real matrices. Let  $\Pi := \Pi_n$  be the set of permutation functions (bijections) of the set  $[n] = \{1, \dots, n\}$ . We define the Koopmans-Beckmann (KB) version QAP via:

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

Note that occasionally an additional linear cost function is added, though we drop it here for brevity.

Equation (1) can be recast in matrix notation. Let  $\mathcal{P}$  be the set of  $n \times n$  permutation matrices  $\mathcal{P} = \{P \in \{0, 1\}^{n \times n} : P^T \mathbf{1} = P \mathbf{1} = \mathbf{1}\}$ , where  $\mathbf{1}$  is the  $n$ -dimensional column vector consisting of all 1’s. Thus, we can write  $PAP^T = (a_{\pi(u)\pi(v)})$  whenever  $P$  is the permutation matrix corresponding to the bijection  $\pi$ , yielding the following equivalent optimization problem:

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

This can be written more compactly in trace notation via:

$$\begin{aligned} \text{(QAP)} \quad & \text{minimize} \quad \text{tr}(B^\top P^\top A P) \\ & \text{subject to} \quad P \in \mathcal{P}. \end{aligned} \tag{2}$$

We hereafter refer to (2) as the QAP optimization function.

## 2.2 Relaxed Quadratic Assignment Problem

Eq. (2) is a binary quadratic program with linear constraints. Because of the combinatorial nature of the feasible region, finding a *global* optimum is **NP**-hard. Rather than directly optimizing over the permutation matrices, we begin by relaxing the constraint set to the convex hull of the set of permutation matrices, the set of doubly stochastic matrices (i.e. the Birkhoff polytope)

$$\mathcal{D} := \mathcal{D}_n = \{P \in \mathbb{R}^{n \times n} : P^\top \mathbf{1} = P \mathbf{1} = \mathbf{1}, P \succeq 0\},$$

where  $\succeq$  indicates an element-wise inequality. Relaxing  $\mathcal{P}$  to  $\mathcal{D}$  in (2) yields the relaxed quadratic assignment problem (rQAP):

$$\begin{aligned} \text{(rQAP)} \quad & \text{minimize} \quad \text{tr}(B^\top P^\top A P) \\ & \text{subject to} \quad P \in \mathcal{D}. \end{aligned} \tag{3}$$

Note that, although rQAP is a quadratic program with *linear* constraints, it is not necessarily convex. Indeed, the objective function,  $f(P) = \text{tr}(B^\top P^\top A P)$ , has a Hessian that is not necessarily positive definite:

$$\nabla^2 f(P) = B \otimes A + B^\top \otimes A^\top,$$

where  $\otimes$  indicates the Kronecker product.

While nonconvex quadratic optimization is, in general, **NP**-hard, relaxing the feasible region allows us to employ the tools of continuous optimization to search for a *local* optimum. These local optima can then be projected onto  $\mathcal{P}$ , yielding an approximate solution of (2). Unfortunately, when relaxed to  $\mathcal{D}$ , the QAP optimization function is often multimodal, making initialization important when solving (3).

## 2.3 Graph Matching

A labeled graph  $G = (\mathcal{V}, \mathcal{E})$  consists of a vertex set  $\mathcal{V} = [n]$ , and an edge set  $\mathcal{E} \subset \binom{\mathcal{V}}{2}$  in the undirected case or  $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$  in the directed case. For an  $n$ -vertex graph  $G$ , we define the associated adjacency matrix  $A = (a_{uv}) \in \{0, 1\}^{n \times n}$  to be the binary  $n \times n$  matrix with

$$a_{uv} = \begin{cases} 1 & \text{if } \{u, v\} \in \mathcal{E}, \text{ and } G \text{ is undirected} \\ 1 & \text{if } (u, v) \in \mathcal{E}, \text{ and } G \text{ is directed} \\ 0 & \text{else} \end{cases}$$

Given a pair of  $n$ -vertex graphs  $G_A = (\mathcal{V}_A, \mathcal{E}_A)$  and  $G_B = (\mathcal{V}_B, \mathcal{E}_B)$ , with respective adjacency matrices  $A$  and  $B$ , we consider the following two closely related problems:

- **Graph Isomorphism (GI):** Does there exist a  $P \in \mathcal{P}$  such that  $A = PBP^\top$ .
- **Graph Matching:**  $\min_{P \in \mathcal{P}} \|A - PBP^\top\|_F$ , where  $\|\cdot\|_F$  is the usual matrix Frobenius norm.

GI is one of the few problems with unknown computational complexity [21]. Indeed, if **P**  $\neq$  **NP**, then GI might reside in an intermediate complexity class called **GI**-complete. Moreover, GI is, at worst, only moderately exponential, with complexity  $\mathcal{O}(\exp\{n^{1/2+o(1)}\})$  [22]. On the other hand, GM—reducible to a

QAP—is known to be **NP**-hard in general. Although polynomial time algorithms are available for GM (and GI) for large classes of problems (e.g., planar graphs, trees) [23], these algorithms often have lead constants which are very 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  $(512k^3)!$  [24].

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.

## 2.4 Graph Matching as a Quadratic Assignment Problem

Given a pair of  $n$ -vertex graphs  $G_A = (\mathcal{V}_A, \mathcal{E}_A)$  and  $G_B = (\mathcal{V}_B, \mathcal{E}_B)$ , with respective adjacency matrices  $A$  and  $B$ , we can formally write the graph matching problem as an optimization problem:

$$\begin{aligned} & \text{minimize} && \|A - PBP^\top\|_F \\ & \text{subject to} && P \in \mathcal{P}. \end{aligned} \tag{4}$$

Simple algebra yields that,

$$\begin{aligned} \|A - PBP^\top\|_F^2 &= \text{tr}\{(A - PBP^\top)^\top(A - PBP^\top)\} \\ &= \text{tr}(A^\top A) + \text{tr}(BB^\top) - 2\text{tr}(APB^\top P^\top). \end{aligned} \tag{5}$$

The GMP is then equivalent (i.e. same argmin) to

$$\begin{aligned} (\text{GM}) \quad & \text{minimize} && -\text{tr}(APB^\top P^\top) \\ & \text{subject to} && P \in \mathcal{P}. \end{aligned} \tag{6}$$

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 QAP approximation algorithms also immediately yields an analogous GM approximation.

As is common in solving general QAPs, GM algorithms often begin by first relaxing (4) to a continuous problem (see, for example, [25]). The resulting problem is a convex quadratic program and can be efficiently exactly solved. The obtained solution is then projected back onto  $\mathcal{P}$  yielding an approximate solution to (4). Contrary to popular existing approaches, our FAQ algorithm, first solves a relaxed version of (6) and subsequently projects the solution back onto  $\mathcal{P}$ . This relaxation yields an indefinite quadratic program, which are **NP**-hard to solve in general. However, recent theory indicates that the indefinite relaxation of (6), and *not* the convex relaxation of (4) is provably correct [26]. Unsurprisingly, FAQ thus obtains state-of-the-art performance in terms of both computational efficiency and objective function value for various QAPs (see Section 4).

## 3 Fast Approximate Quadratic Assignment Problem Algorithm

Our algorithm, called FAQ, proceeds in three steps:

- 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 step.

**A: Find a suitable initial position.** While any doubly stochastic matrix would be a feasible initial point, we choose the noninformative “flat doubly stochastic matrix,”  $J = \mathbf{1} \cdot \mathbf{1}^\top/n$ , i.e. the barycenter of the feasible region.

Alternately, we could use multiple restarts, each initial point near  $J$ . Specifically, we could sample  $K$ , a random doubly stochastic matrix using 10 iterations of Sinkhorn balancing [27], and let  $P^{(0)} = (J + K)/2$ . Given this initial estimate, we would then iterate the following five steps until convergence.

**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 first-order optimization program originally devised to solve convex quadratic programs [5, 28]. Although FW is a relatively standard solver, especially as a subroutine for QAP algorithms [29], 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

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

*Step 2, Compute the search direction  $Q^{(i)}$ :* The search direction is given by the argument that minimizes a first-order Taylor series approximation to  $f(P)$  around the current estimate,  $P^{(i)}$ :

$$\tilde{f}^{(i)}(P) \triangleq f(P^{(i)}) + \nabla f(P^{(i)})^\top (P - P^{(i)}). \quad (7)$$

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

$$\begin{aligned} & \text{minimize} && \nabla f(P^{(i)})^\top P \\ & \text{subject to} && P \in \mathcal{D}. \end{aligned} \quad (8)$$

Let  $Q^{(i)}$  indicate the argmin of Eq. (8). As it turns out, Eq. (8) can be solved as a *Linear Assignment Problem* (LAP). The details of LAPs are well known [30], 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 [31–33]. 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 [30, 34]. 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  $\mathcal{D}$ .

$$\begin{aligned} & \text{minimize} && f(P^{(i)} + \alpha^{(i)}Q^{(i)}) \\ & \text{subject to} && \alpha \in [0, 1]. \end{aligned} \quad (9)$$

Let  $\alpha^{(i)}$  indicate the argmin of Eq. (9). 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

$$P^{(i+1)} = P^{(i)} + \alpha^{(i)}Q^{(i)}. \quad (10)$$

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

$$\begin{aligned} & \text{minimize} && -\langle P^{(imax)}, P \rangle \\ & \text{subject to} && P \in \mathcal{P}. \end{aligned} \quad (11)$$

where  $\langle \cdot, \cdot \rangle$  is the usual Euclidean inner product, i.e.,  $\langle X, Y \rangle \triangleq \text{tr}(X^\top Y) = \sum_{ij} x_{ij}y_{ij}$ . Note that Eq. (11) is a LAP (again, see appendix for details).

Pseudocode 2 provides pseudocode for the whole algorithm.

---

**Algorithm 2** FAQ for finding a local optimum of rQAP
 

---

**Require:** graphs  $A$  and  $B$  as well as stopping criteria

**Ensure:**  $\hat{P}$ , an estimated permutation matrix

- 1: Choose an initialization,  $P^{(0)} = \mathbf{1}\mathbf{1}^T/n$
  - 2: **while** stopping criteria not met **do**
  - 3:   Compute the gradient of  $f$  at the current point via Eq. (7)
  - 4:   Compute the direction  $Q^{(i)}$  by solving Eq. (8) via the Hungarian algorithm
  - 5:   Compute the step size  $\alpha^{(i)}$  by solving Eq. (9)
  - 6:   Update  $P^{(i)}$  according to Eq. (10)
  - 7: **end while**
  - 8: Obtain  $\hat{P}$  by solving Eq. (11) 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 [30, 34], 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 ( $\approx 10^{-9}$  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.

**Figure 1.** Running time of FAQ as function of number of vertices. Data was sampled from an Erdős-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  $\approx 10^{-9}$  seconds. FAQ finds the optimal objective function value in every simulation.

### 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 [35]. Recent graph matching papers typically evaluate the performance of their algorithm on 16 of the benchmarks that are known to be “particularly difficult” [25, 36]. We compare the results of FAQ to the results of four other state-of-the-art graph matching algorithms: (1) the PATH algorithm, which solves a path between a convex and concave relaxation of QAP [25], (2) QCV which is the convex relaxation used to initialize the PATH algorithm, (3) the RANK algorithm [37], which uses a spectral decomposition, and (4) the Umeyama algorithm (denoted by U henceforth), which also uses a spectral decomposition [1]. We chose these four algorithms to compare because the code is freely available from the graphm package [25]. 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  $X \in \{\text{PATH, QCV, RANK, U, all}\}$ , 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.

**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.

### 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 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 they 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.

**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.

### 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 [25]. If FAQ yields a lower objective function value than PATH, and is faster, then it clearly is superior to PATH on such problems. Figure 4 compares 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.

### 4.5 QAP Directed Benchmarks

Recently, Liu et al. [38] 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

**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.

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

## 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  $A = PBP^T$ , so that  $\langle A, PBP^T \rangle = 2m$ , 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  $\langle A, DBD^T \rangle > \langle A, PBP^T \rangle = 2m$  (remember that we are minimizing the negative Euclidean inner product). For that to be the case, it must be that  $(DBD^T)_{uv} \geq 1$  for some  $(u, v)$ . That this is not so may be seen by the submultiplicativity of the norm induced by the  $\ell_\infty$  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 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 [39]), 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 [27], 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 and 100 times, reporting only the best result (indicated by FAQ<sub>3</sub> and FAQ<sub>100</sub>, 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 the subgraph with 279 somatic neurons that form edges with other neurons [40, 41].

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  $A_{uv} \in \{0, 1, 2, \dots\}$  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  $B^{(k)} = Q^{(k)} A Q^{(k)T}$ , 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  $k = 1, 2, \dots, 1000$ , 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 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 symmetrized 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

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

**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.

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 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 [25, 42] and [43], which are all of the “PATH” following variety. Zaslavskiy et al. seems to consider but discard FAQ [25] 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 [44] or as subroutines for finding exact solutions [45]. Our work seems to be the first to utilize the precise algorithm described in Pseudocode 2 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) 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. We are working on combining FAQ with dimensionality reduction procedures to achieve the desired scaling from FAQ.

Additional future work might generalize FAQ in a number of ways:

- Many (brain-) graphs of interest will be errorfully observed [46], 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 [19].
- For many brain-graph matching problems, the number of vertices will not be the same across the brains. Recent work from [25, 42] and [43] suggest that extensions in this direction would be both relatively straightforward and effective.
- Often, a partial matching of the vertices is known a priori, and could modify FAQ to leverage these seeded vertices to match the remaining unseeded vertices.
- 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 [47].

- Our matrices have certain special properties, namely sparsity, which makes faster LAP subroutines [34] and more efficient algorithms (such as “active set” algorithms) readily available for further speed increases.
- In many graph settings, we have some prior information that could easily be incorporated into the GM problem in the form of vertex attributes and features. For example, in brain graphs position in the brain, cell type, etc., could be used to measure “dissimilarity” between vertices and we could incorporate this data into FAQ’s objective function to better match the graphs.
- 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 theoretically justified, fast, effective, and easily generalizable. Our algorithm achieves state-of-the-art matching performance and efficiency on a host of benchmark QAP problems and connectome data sets. 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>. We have further incorporated FAQ (as `sgm.R`) into the open-source R package, `igraph`, available for download at <https://github.com/igraph/xdata-igraph/>.



## 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

$$\begin{array}{ll} \text{(LAP)} & \text{minimize} \quad \langle P, AB^T \rangle \\ & \text{subject to} \quad P \in \mathcal{P}. \end{array} \quad (12)$$

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) [30, 34].

To see that Eq. (8) is identical to Eq. (13), 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:

$$\begin{array}{ll} \text{(rLAP)} & \text{minimize} \quad \langle P, AB^T \rangle \\ & \text{subject to} \quad P \in \mathcal{D}. \end{array} \quad (13)$$

As it turns out, the minima of LAP and rLAP are equal to one another [30]. 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 and Notes

1. Umeyama S (1988) An Eigendecomposition Approach to Weighted Graph Matching Problems. Analysis I.
2. Koopmans TCTC, Beckman M, Beckmann M (1957) Assignment Problems and the Location of Economic Activities. The Econometric Society 25: 53–76.
3. Papadimitriou CH, Steiglitz K (1998) Combinatorial Optimization: Algorithms and Complexity. Dover Publications, 496 pp. URL [http://books.google.com/books?hl=en&lr=&id=u1RmDoJqkF4C&oi=fnd&pg=PR11&dq=Combinatorial+optimization:+algorithms+and+complexity&ots=S8xMKZ2Xyb&sig=\\_bloCXwzNDmQS7XEt3oYM9zVAHc](http://books.google.com/books?hl=en&lr=&id=u1RmDoJqkF4C&oi=fnd&pg=PR11&dq=Combinatorial+optimization:+algorithms+and+complexity&ots=S8xMKZ2Xyb&sig=_bloCXwzNDmQS7XEt3oYM9zVAHc).
4. Burkard RE (2013) The quadratic assignment problem. In: Handbook of Combinatorial Optimization, Springer. pp. 2741–2814.
5. Frank M, Wolfe P (1956) An Algorithm for Quadratic Programming. Naval Research Logistics Quarterly 3: 95–110.

6. Bazaraa MS, Sherali HD, Shetty CM (2013) Nonlinear programming: theory and algorithms. John Wiley & Sons.
7. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3: 1–122.
8. Conte D, Foggia P, Sansone C, Vento M (2004) Thirty Years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 18: 265–298.
9. Sporns O, Tononi G, Kotter R (2005) The Human Connectome: A Structural Description of the Human Brain. *PLoS Computational Biology* 1: e42.
10. Hagmann P (2005) From diffusion MRI to brain connectomics. Ph.D. thesis, Institut de traitement des signaux.
11. Gray Roncal W, Koterba ZH, Mhembe D, Kleissas DM, Vogelstein JT, et al. (2013) MIGRAINE: MRI Graph Reliability Analysis and Inference for Connectomics. *Global Conference on Signal and Information Processing* .
12. Herculano-Houzel S (2012) 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: 10661–8.
13. Kolaczyk E (2010) *Statistical Analysis of Network Data: Methods and Models*. Springer, 398 pp. URL [http://www.amazon.com/Statistical-Analysis-Network-Data-Statistics/dp/144192776X/ref=sr\\_1\\_8?s=books&ie=UTF8&qid=1366769236&sr=1-8&keywords=statistics+networks](http://www.amazon.com/Statistical-Analysis-Network-Data-Statistics/dp/144192776X/ref=sr_1_8?s=books&ie=UTF8&qid=1366769236&sr=1-8&keywords=statistics+networks).
14. Csernansky JG, Wang L, Joshi SC, Ratnanather JT, Miller MI (2004) 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: S56–68.
15. Kubicki M, McCarley R, Westin CF, Park HJ, Maier S, et al. (2007) A review of diffusion tensor imaging studies in schizophrenia. *Journal of Psychiatric Research* 41: 15–30.
16. Calhoun VD, Sui J, Kiehl Ka, Turner JA, Allen Ea, et al. (2011) Exploring the psychosis functional connectome: aberrant intrinsic networks in schizophrenia and bipolar disorder. *Frontiers in psychiatry / Frontiers Research Foundation* 2: 75.
17. Fornito A, Bullmore ET (2012) Connectomic intermediate phenotypes for psychiatric disorders. *Frontiers in psychiatry / Frontiers Research Foundation* 3: 32.
18. Fornito A, Zalesky A, Pantelis C, Bullmore ET (2012) Schizophrenia, neuroimaging and connectomics. *NeuroImage* .
19. Vogelstein JT, Priebe CE (2011) Shuffled Graph Classification: Theory and Connectome Applications. Submitted to *IEEE PAMI* .
20. Duin RPW, Pkalska E, Pkalskab E (2011) The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognition Letters* in press.

KEY: Duin2011

ANNOTATION: From Duplicate 2 ( The dissimilarity space: bridging structural and statistical pattern recognition - Duin, Robert P W; Pkalskab, Elbieta ) undefined

21. Fortin S (1996) The Graph Isomorphism Problem. Technical Report, University of Alberta, Dept of CS .
22. Babali L (1981) Moderately Exponential Bound for Graph Isomorphism. *Fundamentals of Computation Theory* : 34–50.
23. Babali L, Erds P, Selkow SM (1980) RANDOM GRAPH ISOMORPHISM. *SIAM Journal on Computing* 9: 628–635.
24. Chen J (1994) A Linear-Time Algorithm for Isomorphism of Graphs of Bounded Average Genus. *SIAM Journal on Discrete Mathematics* 7: 614.
25. Zaslavskiy M, Bach FR, Vert JpP (2009) A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31: 2227–2242.
26. Lyzinski V, Fishkind DE, Fiori M, Vogelstein JT, Priebe CE, et al. (2014) Graph matching: Relax at your own risk. *arXiv preprint, arXiv:14053133* .
27. Sinkhorn R (1964) A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics* 35: 876–879.
28. Bradley SP, Hax AC, Magnanti TL (1977) *Applied Mathematical Programming*. Addison-Wesley, 716 pp. URL <http://www.amazon.com/Applied-Mathematical-Programming-Stephen-Bradley/dp/020100464X>.
29. Anstreicher KM (2003) Recent advances in the solution of quadratic assignment. *SIAM Journal on Optimization* 9: 27–42.
30. Burkard RE, Dell’Amico M, Martello S (2009) *Assignment Problems*. SIAM, 382 pp. URL <http://books.google.com/books?id=nHIzbApL0r0C&pgis=1>.
31. Kuhn HW (1955) The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2: 83–97.
32. Knig D (1931) Gráfok és Mátrixok. *Matematikai és Fizikai Lapok* 38: 116–119.
33. Egeváry J (1931) Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok* 38: 16–28.
34. Jonker R, Volgenant A (1987) A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38: 325–340.
35. Burkard RE, Karisch SE, Rendl F (1997) QAPLIB A Quadratic Assignment Problem Library. *Journal of Global Optimization* 10: 391–403.
36. Schellewald C, Roth S, Schnörr C (2001) 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, pp. 361–368.
37. Singh R, Xu J, Berger B (2007) Pairwise global alignment of protein interaction networks by matching neighborhood topology. *RESEARCH IN COMPUTATIONAL MOLECULAR BIOLOGY* 4453: 16–31.
38. Liu ZY, Qiao H, Xu L (2012) An Extended Path Following Algorithm for Graph Matching Problem. *IEEE transactions on pattern analysis and machine intelligence* 34: 1451–1456.

39. Anstreicher KM (2009) Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization* : 471–484.
40. White JG, Southgate E, Thomson JN, Brenner S (1986) The structure of the nervous system of the nematode *Caenorhabditis elegans*. *Philosophical Transactions of Royal Society London Series B, Biological Sciences* 314: 1–340.
41. Varshney LR, Chen BL, Paniagua E, Hall DH, Chklovskii DB, et al. (2011) Structural Properties of the *Caenorhabditis elegans* Neuronal Network. *PLoS Computational Biology* 7: 1–41.
42. Zaslavskiy M, Bach FR, Vert JpP (2010) Many-to-Many Graph Matching: A Continuous Relaxation Approach. *Machine Learning and Knowledge Discovery in Databases* 6323: 515–530.
43. Escolano F, Hancock ER, Lozano M (2011) Graph Matching through Entropic Manifold Alignment. *Computer Vision and Pattern Recognition* .
44. Anstreicher KM, Brixius NW (2001) A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming* 89: 341–357.
45. Brixius NW (2000) Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations 1 Introduction. *Management* : 1–20.
46. Priebe CE, Vogelstein JT, Bock DD (2011) Optimizing the quantity/quality trade-off in connectome inference. *Communications in Statistics Theory and Methods* : 7.
47. Boyd S, Vandenberghe L (2004) *Convex Optimization*. Oxford University Press, 727 pp. URL <http://www.amazon.com/Convex-Optimization-Stephen-Boyd/dp/0521833787>.