A Box-Constrained Approach for Hard Permutation Problems

Cong Han Lim CONGHAN@CS.WISC.EDU
Stephen J. Wright SWRIGHT@CS.WISC.EDU
Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53706, USA

Abstract

We describe the use of sorting networks to form relaxations of problems involving permutations of n objects. This approach is an alternative to relaxations based on the Birkhoff polytope (the set of $n \times n$ doubly stochastic matrices), providing a more compact formulation in which the only constraints are box constraints. Using this approach, we form a variant of the relaxation of the quadratic assignment problem recently studied in Vogelstein et al. (2015), and show that the continuation method applied to this formulation can be quite effective. We develop a coordinate descent algorithm that achieves a per-cycle complexity of $O(n^2 \log^2 n)$. We compare this method with Fast Approximate QAP (FAQ) algorithm introduced in Vogelstein et al. (2015), which uses a conditional-gradient method whose per-iteration complexity is $O(n^3)$. We demonstrate that for most problems in QAPLIB and for a class of synthetic QAP problems, the sorting-network formulation returns solutions that are competitive with the FAQ algorithm, often in significantly less computing time.

1. Introduction

A *permutation problem* has the form

$$\min_{\pi \in \mathcal{P}^n} \bar{f}(\pi) \tag{1}$$

where $\mathcal{P}^n$ is the set of all permutations of n objects. Linear and quadratic assignment problems (LAP and QAP, respectively) can be formulated as permutation problems, as can seriation problems (Fogel et al., 2013). Many NP-hard problems are special cases of QAP. It is useful to formulate these problems equivalently in terms of permutation matrices $\Pi \in \mathcal{PM}^n$ ($\mathcal{PM}^n$ is the set of $n \times n$ permutation matrices). For example, the graph matching problem can be expressed as $\min_{\Pi \in \mathcal{PM}^n} \|A - \Pi B \Pi^\top\|_F^2$ where A and B are the adjacency matrices of graphs on n vertices. The Birkhoff polytope $\mathcal{B}^n$ — the convex hull of the set of permutation matrices — is frequently used to formulate a continuous (but not necessarily convex) relaxation of such problems, which takes the following form:

$$\min_{P \in \mathcal{B}^n} f(P). \tag{2}$$

The conventional way to implement such formulations is to represent the set $\mathcal{B}^n$ explicitly via n^2 variables (the components of P), $2n$ linear constraints, and n^2 nonnegativity bounds, as follows:

$$\mathcal{B}^n = \{P \in \mathbb{R}^{n\times n} \,|\, P\mathbf{1} = \mathbf{1},\ P^\top \mathbf{1} = \mathbf{1},\ P \geq 0\}, \tag{3}$$

where $\mathbf{1} = (1, 1, \dots, 1)^\top \in \mathbb{R}^n$. Some algorithms based on this representation rely on a projection step to remain within the feasible region; the expense of this step can be ameliorated by the use of an approximate projection technique such as Sinkhorn-Knopp balancing (Sinkhorn & Knopp, 1967). The constraints can also be incorporated directly within a general optimization framework, such as an active-set or interior-point method. A conditional gradient (Frank-Wolfe) framework can also be used (Vogelstein et al., 2015; Zaslavskiy et al., 2009), at a cost of $O(n^3)$ operations per iteration. It has been noted that maintaining feasibility of the iterates with respect to these constraints is typically the bottleneck in these methods.

In this paper, we follow Lim & Wright (2014) in using a *sorting network* to replace the permutation matrix. A sorting network is a mechanism for sorting n objects by means of a composition of pairwise comparators, where each comparator sorts a single pair of inputs. Sorting networks for n numbers can in principle be composed of $O(n \log n)$ comparators; practical networks contain $O(n \log^2 n)$ comparators. Each comparator can be parametrized by a single scalar x, which takes on the value 1 if it is necessary to switch the two inputs to put them in the correct order, and 0 otherwise. By relaxing the feasible set for each parameter x from the binary set $\{0, 1\}$ to the interval $[0, 1]$, we can

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

parametrize a practical sorting network in terms of m variables (where $m = O(n \log^2 n)$) over the box $[0,1]^m$. We can design a sorting network to effect *any* permutation of n objects by making appropriate choices of the parameters x from the binary set $\{0,1\}^m$. We note too that any choice of parameters from the box $[0,1]^m$ yields a matrix in $\mathcal{B}^n$. In fact, the set of matrices traced out by choosing the sorting network parameters from the box $[0,1]^m$ is a manifold lying within $\mathcal{B}^n$ that contains all permutation matrices. Thus, the sorting network yields a relaxation of Problem (1).

A regularization term $\|x-(1/2)\mathbf{1}\|_2^2$ can be included in the sorting-network relaxation, whose function is to nudge solutions toward vertices of the feasible box $[0,1]^m$, and thus closer to a permutation. By scaling this term with a regularization parameter μ, and gradually altering this parameter from a positive value to a negative one, the problem can be transformed from a convex problem to a coordinatewise strongly concave problem whose global minimizer coincides with the minimum of the original problem. Estimates of Lipschitz constants for the objective provide information about appropriate choices for the parameter. A continuation method can track the solution as this parameter is decreased. (We note that negative values of regularization parameters are slightly atypical, but the use of the terminology "regularization" is still appropriate as this term serves to force the solution toward a more useful value — a value at or near a vertex of the feasible box.)

The sorting-network relaxation is nonconvex in general. However, when the objective f is nonconvex, (as in the graph matching problem, for example), the Birkhoff polytope relaxation is nonconvex also, even though the feasible set $\mathcal{B}^n$ is polyhedral. We note too that any solution of a relaxed problem must be converted via heuristics into an estimate of the solution of the original problem (1). This additional consideration makes it difficult to predict the relative performance of different relaxation methods. The compact, nonlinear relaxations based on sorting-networks may thus be viable alternatives to the simpler but higher-dimensional relaxation resulting from the Birkhoff polytope.

We demonstrate an application of the sorting-network relaxation is to the classic quadratic assignment problem (QAP). Our point of comparison is with the nonconvex *relaxed quadratic assignment problem* formulation from Vogelstein et al. (2015), which uses the Birkhoff polytope and uses conditional gradient approach to solve the resulting indefinite quadratic program. We demonstrate an efficient technique for computing components of the gradient with respect to the sorting-network parameters, with the purpose of using these partial gradients in a cyclic coordinate descent framework. These techniques allow us to perform a full cycle of coordinate descent in complexity $O(nm)$ (that is, $O(n^2 \log^2 n)$ for practical sorting networks).

We demonstrate empirically that using this continuation-based method with the fast cyclic coordinate subroutine can converge to a local minimum significantly faster than the Frank-Wolfe method-based Fast Approximate QAP (FAQ) from (Vogelstein et al., 2015). In particular, we demonstrate on synthetic QAP instances that even with multiple heuristics, the run time of cyclic coordinate descent can be many times faster than FAQ. For quadratic assignment problems from QAPLIB (Burkard et al., 1997) and some other synthetic QAP problems, the cyclic coordinate descent method frequently returns an objective value close to FAQ, with the exception of two problem classes.

It is not our main goal to devise another metaheuristic for QAP, of which there are already a wide variety (see Burkard et al. (2012); Loiola et al. (2007) for recent surveys). Rather, we wish to use this important problem class to demonstrate the effectiveness of sorting-network relaxations for solving certain hard permutation problems.

Outline. We introduce sorting networks in Section 2, describe the mapping ϕ that is the basis of the relaxation, and show how the derivatives of ϕ can be computed efficiently. Section 3 describes the continuation strategy for tracking the solution as a function of the regularization parameter. Section 4 describes the QAP and the approach used to solve it in Vogelstein et al. (2015). We also give details of the cyclic coordinate descent algorithm for solving the sorting-network relaxation. Further implementation details are given in Section 5 and computational results are described in Section 6. The supplementary material contains omitted proofs and details on the experiments.

Related Work. The use of sorting networks in optimization problems involving continuous objectives and permutations as the variables draws on the compact extended formulation of the permutahedron (Goemans, 2015). The extended formulation relies on a "mixing" constraint at each comparator, with parameters to describe the amount of mixing that occurs. Sorting-network relaxations were employed in the seriation problem (which has a convex objective) in Lim & Wright (2014), but the technique of that paper applies only to permutations of vectors. The current paper answers in the affirmative a question raised in Lim & Wright (2014), about whether sorting-network relaxations can be devised for more general problems involving optimization over permutations.

The use of continuation methods for nonconvex problems is popular within the computer vision and machine learning communities, and it has recently been applied to the QAP. The approaches for QAP use the Birkhoff polytope and the fact that the minimum of a concave objective function over any polytope is attained at extreme points. Zaslavskiy et al. (2009) derive convex and concave relaxations over the

Birkhoff polytope for the graph matching problem, which can be used as a proxy for the QAP. They then solve a series of problems over a convex combination of the two problems. Xia (2010) solves a sequence of ℓ_2-regularized problems over the Birkhoff polytope where regularization terms $-\mu\|P\|_F^2$ are added to the objective (2), for different positive values of μ.

2. A Box-Constrained Framework based on Sorting Networks

A sorting network on n elements is a sequence of m *pairwise comparators* $C = \{(\alpha_1,\beta_1),(\alpha_2,\beta_2),\dots,(\alpha_m,\beta_m)\}$ that defines the following sorting algorithm for a sequence of n elements: At the kth round, compare the α_kth and β_kth item in the sequence (where α_k and β_k are both integers in the range $1,2,\dots,n$, with $\alpha_k < \beta_k$) and switch the position of the elements if and only if the β_kth element is smaller. The AKS sorting network (Ajtai et al., 1983) has size $m = O(n\log n)$, and is the most compact option for extremely large n. Practical sorting networks with simple recursive constructions such as the bitonic sorter or odd-even mergesort (Batcher, 1968) have size $m = O(n\log^2 n)$. Sorting networks are frequently depicted as a series of wires and comparators; Figure 1 illustrates a small bitonic network.

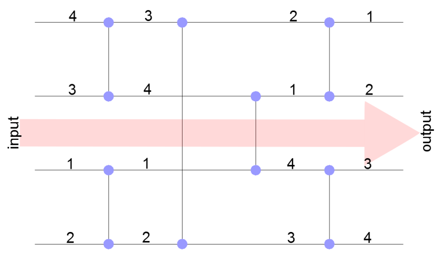

Figure 1. A bitonic sorting network on 4 variables composed of the sequence of comparators $\{(1,2),(3,4),(1,4),(2,3),(1,2),(3,4)\}$. The numbers on the wires illustrate the sorting process for the input sequence $(4,3,1,2)$.

With each comparator (α_k,β_k) in the sorting network we can associate a variable $x_k \in [0,1]$ that defines how the outputs from the comparator are defined as a mixture of the inputs. Using this notation, we can represent this comparator as an elementary matrix $M^k(x_k)$, whose (i,j) element is defined as follows:

$$M_{ij}^k(x_k) := \begin{cases} 1 & i=j,\ i\neq \alpha_k,\beta_k, \\ x_k & i=j,\ i,j\in\{\alpha_k,\beta_k\}, \\ (1-x_k) & i\neq j,\ i,j\in\{\alpha_k,\beta_k\}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Here, we have $M^k(1)=I$ (corresponding to "no swap" of the α_k and β_k elements) whereas $M^k(0)$ is the permutation matrix that "swaps" elements α_k and β_k. The matrix $M^k(x_k)$ is invertible if and only if $x_k \neq 0.5$. Given any $n\times n$ matrix Y, it takes only $O(n)$ operations to update Y to $(M^k(x_k))^{-1}Y$ (if $x_k\neq 0.5$) or $M^k(x_k)Y$, since we only have to update two rows in each case.

The derivative of M^k with respect to its argument is a rank-one matrix $v^k(v^k)^\top$, where

$$v_i^k = \begin{cases} 1 & i=\alpha_k, \\ -1 & i=\beta_k, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The following function $\phi:\mathbb{R}^m\to\mathbb{R}^{n\times n}$ defines the composition of the transformations $M^k(x_k)$ across the full network:

$$\phi(x) := M^m(x_m)M^{m-1}(x_{m-1})\dots M^1(x_1),$$

The image $\{\phi(x)\mid x\in[0,1]^m\}$ includes all permutation matrices, since the sorting network allows us to pick a binary x to represent any permutation. The image is in general a strict subset of the Birkhoff polytope: It is easy to verify that $\phi(x)\mathbf{1}=\mathbf{1}$ and $\phi(x)^\top\mathbf{1}=\mathbf{1}$ for any $x\in[0,1]^m$, and it is also clear that $\phi(x)\geq 0$.

The following property of the mapping ϕ will be useful.

Lemma 2.1. $\|\phi(x)-\phi(y)\|_F \leq 2\|x-y\|_1$.

We can use the mapping ϕ to replace the argument P in the formulation (2). After adding a regularization term $\mu\in\mathbb{R}$, we obtain the following sorting-network-based formulation:

$$\min_{x\in[0,1]^m} g(x;\mu) = f(\phi(x)) + \mu\|x-(1/2)\mathbf{1}\|_2^2. \tag{6}$$

The regularization term μ allows us to develop a *continuation*-based approach in later sections, which works by gradually decreasing μ to transform (6) from convex to concave.

If f has gradients with Lipschitz constant L_f, we can derive Lipschitz constants for the gradient of the composed function $f(\phi)$ by applying Lemma 2.1.

$$\begin{aligned} |\nabla f(\phi(x)) - \nabla f(\phi(y))| &\leq L_f\|\phi(x)-\phi(y)\|_F \\ &\leq 2L_f\|x-y\|_1 \qquad (7) \\ &\leq 2\sqrt{m}L_f\|x-y\|_2. \qquad (8) \end{aligned}$$

Inequality (8) ensures that if $\mu > \sqrt{m}L_f$ or $\mu < -\sqrt{m}L_f$, then Problem (6) is strongly convex or strongly concave respectively. Inequality (7) shows that $|\mu| > L_f$ suffices for *coordinatewise* convexity or concavity, that is, the function restricted to a single coordinate for any coordinate is convex or concave. Coordinatewise concavity guarantees that the global solutions for Problems (1) and (6) coincide.

Proposition 2.2. *Suppose the objective $g(x;\mu)$ in Problem (6) is coordinatewise strongly concave. Then any point $\hat{x}$ that is coordinatewise minimal leads to a permutation matrix $\phi(\hat{x})$. Any $\phi(x^*)$ corresponding to a global minimum x^* is a solution to the original Problem (1).*

3. A Continuation-Based Algorithm for Obtaining a Permutation.

The continuation method, sometimes also known as graduated nonconvexity or the homotopy method, is a well-known technique used to solve difficult problems. Instead of tackling the target problem directly, the method solves a series of subproblems, using the approximate solution of each subproblem to initialize the next one. In our case, the continuation method begins from a convex problem and gradually transforms it to a problem equivalent to the original hard nonconvex problem, the intuition being that the sequence of solutions gradually follows a path to a good solution for the original problem.

To define a continuation-based algorithm, we need to determine the sequence of subproblems and the method for solving each subproblem. In this paper, we use cyclic coordinate descent with exact minimization to solve each subproblem. We terminate the cyclic coordinate descent when the objective value fails to meet a sufficient decrease criteria. The subproblems in our formulation depend on the regularization term μ. We know from Proposition 2.2 that once $\mu < -L_f$, the problem is coordinatewise strongly concave, so that cyclic coordinate descent finds a binary vector x. If we choose to further decrease μ once we have found a binary x, the new solution would not move.

Proposition 3.1. *Suppose $g(\cdot;\mu)$ is coordinatewise strongly concave. Then cyclic coordinate descent with exact minimization reaches a binary x in finitely many steps.*

Proposition 3.2. *Suppose binary vector x is coordinatewise minimal for $g(x;\mu_0)$. For any $\mu < \mu_0$, x is still a coordinatewise minimum for $g(x;\mu)$.*

The proof of Proposition 3.2 follows directly from symmetry of the regularization term around 0.5. Note that any coordinatewise minimal point is also a stationary point, hence no first-order method will find a new point.

By considering both Propositions 3.1 and 3.2, we know that we should not decrease μ beyond $-L_f$ more than once. This gives us a lower bound for μ. Proposition 3.2 also provides an early termination criteria for the algorithm.

Remaining issues are to determine the starting value of μ, and how to update μ between subproblems. For continuation methods, it is typical to set the initial μ to make the problem easy (that is, convex or strongly convex), but empirically we have observed that it good computational results can be obtained by using 0 as the initial value of μ, and decreasing it by a constant value between each subproblem. The approach is specified in Algorithm 1.

Algorithm 1 Continuation Algorithm for (12)

```
Input: $g : [0,1]^m \to \mathbb{R}, x \in \mathbb{R}^m, \mu, c, \mu_{\min} < -L_f - c, \epsilon_{\text{dec}}, \epsilon_{\text{bin}}$
repeat
  repeat
    $x' \leftarrow x$
    $x \leftarrow$ one cycle of coordinate descent on $x'$
  until $g(x';\mu) - g(x;\mu) < \epsilon_{\text{dec}}$
  if $\mu > \mu_{\min}$ then
    $\mu \leftarrow \mu - c$
  else
    break
  end if
until $\|x - \text{round}(x)\|_2 < \epsilon_{\text{bin}}$
return round$(x)$
```

4. A Box-Constrained Relaxation of the QAP and Fast Cyclic Coordinate Descent

The classic quadratic assignment problem (QAP), as formulated by Koopmans and Beckmann, can be written as follows:

$$\min_{\Pi \in \mathcal{P}^n} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi_i \pi_j} = \left\langle A, \Pi B \Pi^\top \right\rangle . \tag{9}$$

This problem is NP-hard and can be used to model problems such as facility location, graph partitioning, the traveling salesman problem, and many other combinatorial optimization problems. There is a wealth of research on this topic, and we refer the reader to some of the comprehensive surveys available (Loiola et al., 2007; Pardalos & Wolkowicz, 1994).

In this paper, we focus on a particular continuous relaxation to the problem studied by Vogelstein et al. (2015), who define the following relaxation over the Birkhoff polytope:

$$\min_{P \in \mathcal{B}^n} \left\langle A, PBP^\top \right\rangle = \langle AP, PB \rangle . \tag{10}$$

This formulation is referred to in Vogelstein et al. (2015) as the *indefinite relaxed QAP* formulation. Its nonconvex quadratic objective makes finding the global minimum an intractable task, in general. Vogelstein et al. (2015) introduce a Frank-Wolfe-based *fast approximate QAP* (FAQ) algorithm for finding local minima, in which each iteration requires solution of a linear assignment problem, an $O(n^3)$ operation. The Hessian of the objective is given by the sum of Kronecker produts $B \otimes A + A^\top \otimes B^\top$, so the Lipschitz constant associated with the gradients is given by

$$L_{\text{rQAP}} = \max\left\{\{|\lambda_A^{\min}|, |\lambda_A^{\max}|\} \cdot \{|\lambda_B^{\min}|, |\lambda_B^{\max}|\}\right\} \tag{11}$$

where $\cdot$ denotes the elementwise product of the sets and $\lambda_U^{\min}$ and $\lambda_U^{\max}$ are the smallest and largest eigenvalues of U respectively.

The sorting-network formulation corresponding to (10) is

$$\min_{x \in [0,1]^m} \langle A\phi(x), \phi(x)B\rangle + \mu\|x - (1/2)\mathbf{1}\|_2^2. \tag{12}$$

This formulation has a multiquadratic objective function: The exponent for each variable in each term is at most 2. To our knowledge, box-constrained multiquadratic programming has not been studied in the literature. From equations (7) and (11) we know that setting $|\mu| > L_{\text{rQAP}}$ makes this formulation coordinatewise strongly convex or concave.

Evaluating the objective function takes $O(nm)$ time for this problem, because each of the m multiplications by matrices $M^k(x_k)$, $k = 1, 2, \dots, m$ required to form $A\phi(x)$ and $\phi(x)B$ takes $O(n)$ operations, while the inner product in (12) requires $O(n^2)$. Computing the derivative with respect to an arbitrary coordinate can take up to $O(nm)$ operations. However, as we describe below, the form of the objective function can be exploited to compute one complete cycle of the cyclic coordinate descent method in $O(nm)$ time. (In the supplementary material, we show how this structure also allows us to evaluate a full gradient in $O(nm)$ operations, and the Hessian matrix in $O(m^2)$ operations.)

A fast implementation of the cyclic coordinate descent method for (12)**.** The cyclic coordinate descent method (sometimes known as the Gauss-Seidel method) performs the following update to the kth coordinate on the ith cycle:

$$x_k^{i+1} \leftarrow \operatorname*{argmin}_{\xi \in [0,1]} g(x_1^{i+1}, \dots, x_{k-1}^{i+1}, \xi, x_{k+1}^i, \dots, x_m^i; \mu).$$

There are a few reasons to adopt a cyclic coordinate descent approach for our problem over other first-order methods. First, the multiquadratic form of the objective (12) means that the function is simply a quadratic along each coordinate. Once we obtain the coefficients that define the quadratic, we can efficiently find the exact minimizer. Second, the form of the function ϕ allows us to determine the coefficients of the quadratic incrementally and efficiently.

Recalling the definition (4), we can express the objective of (6) (when $\mu = 0$) as follows:

$$\left\langle AM^m(x_m)\dots M^1(x_1), M^m(x_m)\dots M^1(x_1)B\right\rangle$$

(For notational convenience, we omit the argument from the $M^k(x_k)$ terms when x_k can be inferred.) We will compute the coefficients of the quadratic form along each coordinate from the first and second derivatives. Using (5), we write the gradient with respect to x_k as the sum of the following two terms:

$$\left\langle AM^m \dots M^{k+1}v^k(v^k)^\top M^{k-1}\dots M^1, \phi(x)B\right\rangle, \tag{13}$$

$$\left\langle A\phi(x), M^m \dots M^{k+1}v^k(v^k)^\top M^{k-1}\dots M^1 B\right\rangle, \tag{14}$$

while the second derivative with respect to x_k is

$$\begin{aligned} 2 \cdot \big\langle (v^k)^\top M^{k+1}\dots M^m AM^m \dots M^{k+1}v^k,\\ (v^k)^\top M^{k-1}\dots M^1 BM^1 \dots M^{k-1}v^k\big\rangle. \end{aligned} \tag{15}$$

We now demonstrate how to compute the coordinatewise first and second derivatives incrementally, coordinate by coordinate. Consider (13) (the analysis for (14) is similar). After rearranging, and using such properties of the inner product as $\langle AB, C\rangle = \langle A, CB^\top\rangle$, this term equals

$$\begin{aligned} \big\langle M^{k+1}\dots M^m AM^m \dots M^{k+1}v^k,\\ M^k(x_k)M^{k-1}\dots M^1 BM^1 \dots M^{k-1}v^k\big\rangle, \end{aligned} \tag{16}$$

which by defining

$$\begin{aligned} S^k &:= M^{k+1}M^{k+2}\dots M^m AM^m \dots M^{k+2}M^{k+1}\\ T^k &:= M^{k-1}M^{k-2}\dots M^1 BM^1 \dots M^{k-2}M^{k-1}, \end{aligned}$$

can be written as

$$\langle S^k v^k, M^k(x_k)T^k v^k\rangle. \tag{17}$$

After updating x_k to x_k' and using $M^k := M^k(x_k')$ in the definition of T^{k+1}, we can write the analogue of (17) for the next coordinate $k+1$ as follows:

$$\left\langle S^{k+1}v^{k+1}, M^{k+1}(x_{k+1})T^{k+1}v^{k+1}\right\rangle \tag{18}$$

Suppose $x_{k+1} \neq 0.5$. To compute (18) after computing (16), we need to compute $S^{k+1} = (M^{k+1})^{-1}S^k(M^{k+1})^{-1}$ and $T^{k+1} = M^k T^k M^k$ (both $O(n)$ operations), perform several other $O(n)$ matrix-vector operations, and finally form an inner product (also $O(n)$). Note that the initial value $S^0 = M^1M^2\dots M^m AM^m \dots M^2M^1$ requires $O(nm)$ operations to compute, a similar cost to the subsequent sweep through the coordinates.

If $x_{k+1} = 0.5$, we can no longer apply the above procedure since the matrix M^{k+1} is not invertible. We can circumvent this issue by storing explicitly the columns of $AM^m \dots M^{k+1}$ that differ from $AM^m \dots M^k$. In practice, we would also store the columns instead of computing the inverse when x_{k+1} is very close to 0.5 to avoid numerical issues. Similar procedures to those just described can be used to compute the second term (14) and the Hessian (15). We summarize the approach in Algorithm 2, and the iteration complexity is described in the following result.

Algorithm 2 A cycle of cyclic coordinate descent for (12)

```
Input: comparison coefficients $x$, matrices $A, B \in \mathbb{R}^{n\times n}$ and regularization coefficient $\mu$
$S \leftarrow A$
for $i = m$ to $1$ do
  $S \leftarrow M^i S M^i$
end for          // $S = \phi(x)^\top A \phi(x)$ at the end of the loop
$T \leftarrow B$
for $i = 1$ to $m$ do
  $S \leftarrow (M^i)^{-1} S (M^i)^{-1}$
  $a \leftarrow 2 \cdot \langle (v^i)^\top S v^i, (v^i)^\top T v^i \rangle$
  $b \leftarrow \langle S v^i, M^i T v^i \rangle + \langle (v^i)^\top S, (v^i)^\top T M^i \rangle$
  $a \leftarrow a + \mu, b \leftarrow b + \mu(x - 0.5)$
  $x_i \leftarrow \arg\min_y \left\{ \frac{1}{2} a y^2 + (b - a x_i) y \;\middle|\; y \in [0,1] \right\}$
  $M^i \leftarrow M^i(x_i)$
  $T \leftarrow M^i T M^i$
end for
```

Theorem 4.1. *Each cycle of the cyclic coordinate descent method can be performed in $O(nm)$ time. The amount of additional memory used is $O(n^2 + nd)$, where $d = |\{i \mid x_i = 0.5\}|$.*

In general there is no convergence guarantee for cyclic coordinate descent with exact minimization for nonconvex problems — Powell (1973) provides a well-known example in $\mathbb{R}^3$ where all the limit points of the iterates of cyclic coordinate descent are not stationary points. Nonetheless, we chose to use this over variants with guarantees like random coordinate descent (Patrascu & Necoara, 2014), or cyclic coordinate descent with a proximal term (Grippo & Sciandrone, 2000) due to the better empirical performance.

5. Additional Implementation Details

We supplement the basic approach of Algorithm 2 with various other features and heuristics.

Choice of sorting network. As with Lim & Wright (2014), we find that using bitonic sorting networks (Batcher, 1968) in our experiments gives us good empirical performance. These sorting networks have a size of $O(n \log^2 n)$ and can be constructed using a simple recursive algorithm in $O(n \log^2 n)$ time.

In addition, having additional comparators in a sorting network may allow us to make additional swaps that improve the objective value. We have found in our experiments that randomly adding comparators can significantly improve performance. We choose index pairs (α_k, β_k), $k = 1, 2, \dots, m'$ at random from $\{1, 2, \dots, n\} \times \{1, 2, \dots, n\}$ and append these pairs to the list of comparators defined by the bitonic network. There is a trade-off in performance improvement and running time when adding comparators. We found that setting the number m' of new comparators to m works well in practice.

Multiple restarts. The nonconvexity of formulations (9) and (12) means that both methods often terminate at stationary points which could be far from the global optimum. To improve the quality of the local solutions that are identified, we restart the methods multiple times with different starting points. (Vogelstein et al. (2015) employ this technique, with good results.) We can also permute the rows and columns of A differently for each restart of the algorithm, since the behavior of the function $\phi(x)$ is not invariant to such rearrangements.

Finding a permutation that is locally optimal (up to a single swap) using coordinate descent. The following method ensures that the permutation $\phi(x)$ computed by our method is at least as good as all permutations that differ from $\phi(x)$ by at most one swap. After obtaining a permutation, instead of directly terminating the continuation algorithm, we rearrange the rows and columns of A and B according to this permutation. We then define a sorting network in which the list of comparators consists of every pair of elements from $\{1, 2, \dots, n\}$, and set $x = \mathbf{1}$. We then resume the continuation algorithm using this new sorting network. If the cyclic coordinate descent algorithm results in no change to the initial value $x = \mathbf{1}$, we know that no single swap could have improved the objective, so we can terminate safely. Otherwise, we repeat this process.

Using this 'full' sorting network of size $O(n^2)$ may be slow, and we can cap the number of iterations with this network to ease the computational burden. Alternatively, we can generate a smaller random sorting network by choosing m' index pairs at random (m' controls the trade-off between local optimality and runtime) and forming a composition of the comparators defined by these pairs. (The resulting list of comparators is not strictly speaking a sorting network, since it is not guaranteed to sort all inputs.)

6. Numerical Results

Vogelstein et al. (2015) demonstrate that FAQ outperforms other approximate QAP methods, so we focus on comparing our continuation-based approach against the FAQ algorithm on quadratic assignment problems from the literature. We report on both the objective values attained and the running time.

Setup. The experiments were run in MATLAB 8.4, limited to a single thread. For the FAQ algorithm, we used the MATLAB implementation provided by Vogelstein et al. (2015) at `github.com/jovo/FastApproximateQAP`, and used the default stopping criteria provided there. Since the runtime for FAQ algorithm is sometimes long, and it often does not make much progress after a certain point, we capped the number of Frank-Wolfe iterations at 10 or 30. (We denote these two variants by FAQ10 and FAQ30, respectively.) The FAQ codes did not converge within their limits of 10 and 30 iterations, but the objective did not improve significantly beyond 30 iterations.

Heuristics or other local search methods that were developed for QAP can improve the performance of these algorithms, but a detailed analysis of how to compose heuristics is beyond the scope of this paper. In this paper we only consider a simple *greedy pairwise swap* heuristic. In this approach, we iterate through all possible pairwise swaps until we find a swap that improves the objective, then perform that swap. In the main paper we consider FAQ10+G/FAQ30+G, which applies the following greedy pairwise swap heuristic up to 30 times after FAQ10/FAQ30 terminates.

We implemented the continuation method with the fast cyclic coordinate descent algorithm in MATLAB. (A straightforward translation of the code into MATLAB/MEX gives us a significant speed-up, but we keep the code in pure MATLAB for a fairer timing comparison with the MATLAB implementation of FAQ.)

We consider multiple variants of coordinate descent with different features/heuristics applied, and collectively refer to them as CD. We set $\epsilon_{\text{dec}} = 0.1\%$ and $\epsilon_{\text{bin}} = 0.1\sqrt{n}$ in Algorithm 1. The features we consider are continuation (C), sorting network heuristic for local optimality (RS and FS, for random and full sorting network, respectively), and the greedy swap heuristic from above (G). For simplicity, we focus primarily in the main paper on CD+C+RS: Coordinate descent with continuation and random sorting networks of size $O(n \log^2 n)$. We decrease μ by $L_{\text{rQAP}}/10$ between subproblems (see (11) for the definition of L_{rQAP}), and terminate after solving the problem for two successive values of μ less than $-L_{\text{rQAP}}$. In each run, we let the algorithm apply the new sorting network heuristic up to 3 times. The use of random sorting networks instead of full-sized sorting networks (with $O(n^2)$ comparators) provides an objective value that is very slightly worse, with significantly better scaling with n.

QAPLIB Problems. We compare the objective values and timings achieved by FAQ and CD on problems from QAPLIB (Burkard et al., 1997), the canonical library of QAP problems, which has 137 problems with sizes varying from $n = 10$ to $n = 256$. Each method was run 100 times, and we compare the objective values corresponding to the best result and the median. We take the best or median solution obtained out of 100 runs for each method and compare them to the global optimal or best known solution in the literature. We compute the optimality gap, the difference between the objective value obtained and the best known solution normalized by the best known solution. We also compare the difference in average running time.

In the main paper, we focus on comparing CD+C+RS with FAQ10, FAQ30, and FAQ30+G, as summarized in Table 1. We also briefly discuss FAQ10+G and other variants of CD.

We focus first on the objective values obtained. Overall, for a majority of the instances the objective values obtained by CD and the FAQ methods are within a few percent of each other. CD+C+RS is better than FAQ10 for 12 families, FAQ30 for 6 families, and FAQ30+G for one family. The performance of FAQ improves significantly when the maximum number of iterations is increased from 10 to 30, and again when the greedy swap heuristic is applied. In particular, while CD+C+RS significantly outperforms FAQ10 and FAQ30 on `chr`, `els`, the greedy swap heuristic allows FAQ30+G to attain a performance close to or better than CD+C+RS. We note that CD+C+RS outperforms FAQ10+G for 5 families. Except for `lipa b` and `ste`, CD+C+RS always provides a gap that is close to or better than the other methods.

For the running time, CD+C+RS generally does much better. Besides beating FAQ10 on a majority of instances and FAQ30/FAQ30+G for all instances, CD+C+RS is never worse by more than a few times, whereas FAQ10 and FAQ30 can be more than 100 times slower (see the `bur` and `els` problems). FAQ10 is approximately three times faster than FAQ30, and the time difference between FAQ10 and FAQ10+G is roughly the same as the difference between FAQ30 and FAQ30+G.

We briefly describe a few observations of the other CD variants. Using the full sorting network compared to a random $O(n \log^2 n)$ network in the heuristic leads to a slightly better objective value in terms of objective value without significantly affecting overall running times. (At small n the choice of new sorting networks has little or negligible effect on running time.) Applying the greedy swap heuristic to CD makes a much smaller difference if the new sorting network heuristic is used. Applying the greedy swap heuristic is often much more time-consuming than applying the new sorting network heuristic for these instances.

Synthetic QAP Problems. To understand better how the running times and objective values scale with problem size, we turn to the QAP generators described in Taillard (1995). These problems have A matrices are Euclidean distance matrices and B matrices are randomly generated with many zero elements. (Further details appear in the supplementary material.) For $n = 300, 400, \ldots, 1000$, we generated one instance each of size n and ran each algorithm 10 times

			Average Percent Optimality Gap								Average Timing Ratio		
			Best of 100 Runs				Median of 100 Runs				(compared with CD+C+FS)		
Type	Num	n	CD+C+RS	FAQ10	FAQ30	FAQ30+G	CD+C+RS	FAQ10	FAQ30	FAQ30+G	FAQ10	FAQ30	FAQ30+G
BUR	8	26	.12	.08	.06	.04	.78	.27	.19	.08	724.04	1895.26	1897.04
CHR	14	12--25	6.84	18.72	9.32	5.90	39.92	77.50	57.76	33.91	.91	1.99	2.38
ELS	1	19	4.21	25.89	22.96	4.56	27.60	77.60	67.39	19.15	394.71	956.55	957.50
ESC	20	16--128	.08	.73	.05	.00	2.02	9.82	5.34	2.36	.37	1.06	1.24
HAD	5	12--20	.00	.39	.15	.05	.39	3.77	.77	.37	1.02	3.09	3.39
KRA	3	30--32	1.75	2.46	.06	.00	5.13	7.28	5.20	4.62	2.67	9.03	9.42
LIPA A	8	20--90	.95	1.79	1.02	.66	1.50	2.23	1.76	1.35	.28	1.80	2.50
LIPA B	8	20--90	7.92	.00	.00	.00	19.03	15.07	4.93	4.90	2.23	1.60	1.98
NUG	15	12--30	.43	.62	.03	.03	3.46	5.04	2.35	1.85	1.29	3.72	3.98
ROU	3	12--20	.22	1.58	.49	.48	4.67	8.45	4.68	3.74	.74	2.19	2.39
SCR	3	12--20	.00	3.94	2.66	.24	7.04	19.36	15.24	8.16	.88	2.46	2.75
SKO	13	42--100	1.03	.94	.34	.28	2.18	2.18	1.23	1.15	2.01	5.29	6.57
STE	3	36	3.63	2.75	1.10	.53	14.77	13.37	8.78	7.80	2.25	6.13	6.65
TAI	28	10--256	1.30	2.90	1.53	1.08	5.50	9.23	6.01	4.71	17.94	66.33	67.48
THO	3	30--150	1.54	1.37	.38	.35	3.62	3.39	2.04	1.88	2.47	6.95	8.23
WIL	2	50,100	.48	.54	.13	.09	1.20	1.39	.61	.51	2.38	6.47	8.35

Table 1. Performance of CD+C+RS, FAQ10, FAQ30 and FAQ+G on QAPLIB instances, aggregated by family of instances. For each family, we take the average optimality gap and the average ratio of the timing of each method to CD+C+RS. The highlighted cells indicates cases where CD+C+RS outperforms that entry.

on each instance. In Figure 2 we compare CD+C+RS, FAQ10+G, and FAQ30+G in terms of the average run time and the gap between the average objective value obtained and the best objective value out of all runs of FAQ30+G.

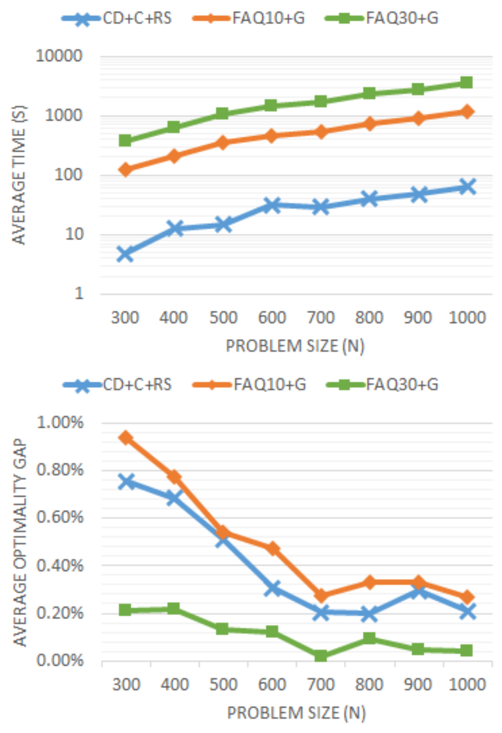

Figure 2. Comparison of CD+C+RS, FAQ10+G, and FAQ30+G for synthetic QAP instances.

For the objective values, FAQ10+G performs worst, followed by CD+C+RS, with FAQ30+G best. The CD+C+RS algorithm is much faster, by a factor of about 20 over FAQ10+G and about 60 over FAQ30+G. Using a full sorting network heuristic has worse scaling effects in terms of the run time. CD+C+FS and CD+FS scale at a rate that is approximately $O(n^3)$, which is the complexity of one cycle of coordinate descent on a $O(n^2)$ sorting network, while the other methods scale similarly at a rate that is closer to $O(n^2)$.

7. Conclusion

We have introduced a novel, compact relaxation for hard permutation problems that reduces the constraints to simple box constraints. These constraints are significantly easier to enforce than those encountered in the Birkhoff polytope relaxation. By incorporating a regularization term, we can gradually alter the problem to a coordinatewise concave problem, where the global minimizer coincides with the original problem.

Using the QAP as an example, we show that the structure of the resulting parametrization can be leveraged to derive fast cyclic coordinate descent algorithms. Our algorithm takes only $O(n)$ time per coordinate or $O(n^2 \log^2 n)$ per cycle, which has better per-iteration complexity than existing methods over the Birkhoff polytope. We develop a continuation-based method for finding a permutation using the new parametrization. Empirical results show that for many problems from QAPLIB, this algorithm can find solutions of quality comparable with the approach of Vogelstein et al. (2015), and significantly faster. A natural goal would be extend this relaxation to other hard permutation problems of interest.

8. Acknowledgements

We thank the anonymous referees for feedback that improved the paper and its presentation. This project was supported by NSF Awards DMS-1216318 and IIS-1447449, Award EM02076 from Exxon Mobil Corp., ONR Award N00014-13-1-0129, AFOSR Award FA9550-13-1-0138, and Subcontract 3F-30222 from Argonne National Laboratory.

References

Ajtai, M., Komlós, J., and Szemerédi, E. An O(n log n) sorting network. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC '83)*, pp. 1–9, New York, New York, USA, December 1983. ACM Press. ISBN 0897910990. doi: 10.1145/800061.808726.

Anstreicher, Kurt M. and Brixius, Nathan W. A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming*, 89(3):341–357, February 2001. ISSN 0025-5610. doi: 10.1007/PL00011402.

Batcher, K. E. Sorting networks and their applications. In *AFIPS '68 (Spring): Proceedings of the Spring Joint Computer Conference*, pp. 307–314, New York, New York, USA, April 1968. ACM Press. doi: 10.1145/1468075.1468121.

Burkard, R., Dell'Amico, M., and Martello, S. *Assignment Problems*. Society for Industrial and Applied Mathematics, 2012. doi: 10.1137/1.9781611972238.

Burkard, Rainer E., Karisch, Stefan E., and Rendl, Franz. QAPLIB A Quadratic Assignment Problem Library. *Journal of Global Optimization*, 10(4):391–403, June 1997. ISSN 0925-5001. doi: 10.1023/A:1008293323270.

Fogel, Fajwel, Jenatton, Rodolphe, Bach, Francis, and D'Aspremont, Alexandre. Convex Relaxations for Permutation Problems. In *Advances in Neural Information Processing Systems*, pp. 1016–1024, 2013.

Goemans, Michel. Smallest compact formulation for the permutahedron. *Mathematical Programming, Series B*, 153(1):5–11, 2015.

Gold, S. and Rangarajan, A. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, April 1996. ISSN 01628828. doi: 10.1109/34.491619.

Grippo, L. and Sciandrone, M. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26(3): 127–136, April 2000. ISSN 01676377. doi: 10.1016/S0167-6377(99)00074-7.

Lim, Cong Han and Wright, Stephen. Beyond the Birkhoff Polytope: Convex Relaxations for Vector Permutation Problems. In *Advances in Neural Information Processing Systems*, pp. 2168–2176, 2014.

Liu, Zhi-Yong, Qiao, Hong, and Xu, Lei. An Extended Path Following Algorithm for Graph Matching Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1451–1456, January 2012. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.45.

Loiola, Eliane Maria, de Abreu, Nair Maria Maia, Boaventura-Netto, Paulo Oswaldo, Hahn, Peter, and Querido, Tania. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657 – 690, 2007. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2005.09.032.

Pardalos, P.M. and Wolkowicz, H. *Quadratic Assignment and Related Problems: DIMACS Workshop, May 20-21, 1993*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1994. ISBN 9780821870624.

Patrascu, Andrei and Necoara, Ion. Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization. *Journal of Global Optimization*, 61(1):19–46, February 2014. ISSN 0925-5001. doi: 10.1007/s10898-014-0151-9.

Powell, M. J. D. On search directions for minimization algorithms. *Mathematical Programming*, 4(1):193–201, dec 1973. ISSN 0025-5610. doi: 10.1007/BF01584660.

Schellewald, Christian, Roth, Stefan, and Schnrr, Christoph. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In Radig, Bernd and Florczyk, Stefan (eds.), *Pattern Recognition*, volume 2191 of *Lecture Notes in Computer Science*, pp. 361–368. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42596-0. doi: 10.1007/3-540-45404-7_48.

Sinkhorn, Richard and Knopp, Paul. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Taillard, Éric D. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3(2): 87–105, August 1995. ISSN 09668349. doi: 10.1016/0966-8349(95)00008-6.

Umeyama, S. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5): 695–703, 1988. ISSN 01628828. doi: 10.1109/34.6778.

Vogelstein, Joshua T., Conroy, John M., Lyzinski, Vince, Podrazik, Louis J., Kratzer, Steven G., Harley, Eric T., Fishkind, Donniell E., Vogelstein, R. Jacob, and Priebe, Carey E. Fast approximate quadratic programming for graph matching. *PloS one*, 10(4), January 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0121002.

Xia, Yong. An efficient continuation method for quadratic assignment problems. *Computers & Operations Research*, 37(6):1027–1032, jun 2010. ISSN 03050548. doi: 10.1016/j.cor.2009.09.002.

Zaslavskiy, Mikhail, Bach, Francis, and Vert, Jean-Philippe. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–42, December 2009. ISSN 1939-3539. doi: 10.1109/TPAMI.2008.245.