Conic Approximation with Provable Guarantee for Traffic Signal Offset Optimization

Yi Ouyang, Richard Y. Zhang, Javad Lavaei, and Pravin Varaiya

***Abstract*—We consider the offset optimization problem that coordinates the offsets of signalized intersections to reduce vehicle queues in large-scale signalized traffic networks. We adopt a recent approach that transforms the offset optimization problem into a complex-valued quadratically-constrained quadratic program (QCQP). Using the special structure of the QCQP, we provide a $\pi/4$-approximation algorithm to find a near-global solution based on the optimal solution of a semidefinite program (SDP) relaxation. Although large-scale SDPs are generally hard to solve, we exploit sparsity structures of traffic networks to propose a numerical algorithm that is able to efficiently solve the SDP relaxation of the offset optimization problem. The developed algorithm relies on a tree decomposition to reformulate the large-scale problem into a reduced-complexity SDP. Under the practical assumption that a real-world traffic network has a bounded treewidth, we show that the complexity of the overall algorithm scales near-linearly with the number of intersections. The results of this work, including the bounded treewidth property, are demonstrated on the Berkeley, Manhattan, and Los Angeles networks. From numerical experiments it is observed that the algorithm has a linear empirical time complexity, and the solutions of all cases achieve a near-globally optimal guarantee of more than 0.99.**

I. Introduction

The operation of urban traffic networks depends on the control of signalized intersections. One key component in traffic signal control is the selection of signal offsets among signalized intersections. The intersection-to-intersection offsets determine the relative timing of signals in a traffic network. An appropriate selection of offsets can align the signals to reduce vehicle idling time at red lights, hence decreases traffic queues and delays in the entire network. Offset optimization seeks to determine signal offsets to coordinate intersections to optimize certain objectives of a traffic network.

One approach to offset optimization is to assume that every link of the network has a link delay function that maps the signal offset of the two ends of the link to the delay incurred at the link. Then, the offset optimization problem is formulated as minimizing the sum of all link delay functions of the network. Under some restrictions on the link delay functions or on the network topology, methods based on dynamic programming [1], [2] and mixed-integer programming formulations [3–5] can be used to solve the offset optimization problem. In addition to certain restrictions, these methods generally suffer computational constraints when the size of the network increases.

Another approach to selecting signal offsets is to formulate the problem as minimizing the queue lengths of all intersections of the network. Regarding this approach, [6] approximates the traffic flow processes as sinusoidal functions of time. With the sinusoidal approximation, the offset optimization problem can be transformed into a complex-valued quadratically-constrained quadratic program (QCQP). It is known that a nonconvex QCQP can be relaxed to a convex semidefinite program (SDP) for which local search algorithms can be deployed to find a global solution of the relaxation efficiently [7]. For smaller networks where the SDP relaxation is exact (i.e., the SDP relaxation has the same optimal objective value as the QCQP), [6] shows that the offsets generated from this sinusoidal approximation achieve a better simulation performance compared to the results based on the link delay function approach.

For large-scale networks with hundreds or thousands of intersections, the SDP relaxation proposed in [6] can be cumbersome to solve in practice due to the time and memory complexities. Furthermore, the SDP relaxation is not exact for many real-world networks. The work [8] uses a Burer-Monteiro method as a heuristic to mitigate the complexity of SDP for large-scale traffic networks. The Burer-Monteiro method is a low complexity nonconvex optimization method for SDP. Although this method leads to a low complexity offset optimization algorithm, it only provides a local optimal solution to the SDP relaxation. Therefore, the performance of the offsets generated from this method may be arbitrarily far from that of the solution of the QCQP for offset optimization.

In this paper, we consider a convex optimization approach to approximately solve the QCQP formulation proposed in [6] for the offset optimization problem. The QCQP can be transformed into a complex-valued convex quadratic maximization problem. This formulation is very similar to the classic MAX-CUT problem in combinatorial optimization [9]. Inspired by the Goemans–Williamson algorithm for MAX-CUT [10], we provide a $\pi/4$-approximation algorithm based on the solution of an SDP relaxation of the QCQP. We then focus on optimally solving the SDP relaxation for large-scale networks to obtain offsets with near-optimal performance.

Although large-scale SDP is generally hard to solve ($O(N^{6.5})$ complexity with N denoting the number of rows in the variable matrix), the SDP relaxation for real-world traffic networks posses some nice sparsity structures. In particular, the sparsity pattern of the SDP relaxation is determined by the topology of the traffic network, which is almost planner and sparse. We propose an optimization algorithm for the SDP relaxation based on the tree conversion technique [11].

Y. Ouyang, R. Y. Zhang, J. Lavaei are with the Department of Industrial Engineering and Operations Research, University of California, Berkeley. P. Varaiya is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Email: {ouyangyi, ryz, lavaei, varaiya}@berkeley.edu. J. Lavaei is also with the Tsinghua-Berkeley Shenzhen Institute, University of California, Berkeley.

This work was supported by the ONR grants N00014-17-1-2933 and N00014-15-1-2835, DARPA grant D16AP00002, AFOSR grant FA9550- 17-1-0163, and ARO grant W911NF-17-1-0555.

Similar methods have shown success in solving large-scale power networks [12–14] and in machine learning [15]. Using a tree decomposition, we first reformulate the SDP relaxation into a reduced-complexity SDP. We then adopt a dualization technique recently developed in [16] to solve the SDP using a general purpose interior-point method. The SDP solution is then used in the $\pi/4$-approximation algorithm to generate near-optimal offsets for the offset optimization problem.

When a network is "sparse" that it has a bounded treewidth [17], we show that the complexity of the underlying algorithm scales near-linearly $O(N^{1.5})$ in time and linearly $O(N)$ in memory with the number of intersections. The bounded treewidth assumption is expected to be satisfied for real-world traffic networks for the following reasons. Most traffic networks are planar graphs by construction because the majority of roads do not cross without intersecting. The treewidth of a planar graph grows at most as the square root of the number of nodes and the worst case is attained by grid graphs. While local networks tend to resemble grids, inter-area traffic systems connecting larger regions are more tree-like, which have small treewidths. Therefore, the treewidth of a large-scale traffic network is normally bounded by the square-root of the size of its largest grid which is relatively small in practice. By analyzing different real-world traffic networks, we show that the treewidths are indeed small even for large city areas such as Manhattan and Downtown Los Angeles. Through numerical experiments, we verified that the proposed algorithm has a linear empirical time complexity for networks with up to twelve thousand intersections. Furthermore, the solutions of all studied cases achieve a global performance guarantee of at least 0.99.

Notation

The sets of real and complex numbers are denoted by $\mathbb{R}$ and $\mathbb{C}$. The notation "i" is the imaginary unit. For a complex number $x \in \mathbb{C}$, Re(x), Im(x), $\bar{x}$, $\angle x$, and $|x|$ denote the real part, imaginary part, conjugate, angle, and absolute value of the number x. I is the identity matrix and $\mathbf{1}$ denotes the vector-of-ones; their sizes are inferred from context. For a vector $v \in \mathbb{C}^N$, $v[j]$ denotes the j-th element of v and v^H is the conjugate transpose of v for $j = 1, \ldots, N$. For a matrix $X \in \mathbb{C}^{N\times N}$, $X[j,k]$ denotes the (j,k)-th element of X and $X[\mathcal{I}_j, \mathcal{I}_k]$ indicates the submatrix of X with the row and column index sets $\mathcal{I}_j$ and $\mathcal{I}_k$, respectively. Moreover, $\mathrm{tr}(X)$ and $\mathrm{rank}(X)$ denote the trace and rank of X. Also, X^H denotes the conjugate transpose, and $\mathrm{vec}(X)$ denotes the vectorization of X. $X \succeq 0$ means that X is Hermitian and positive semidefinite. $|\mathcal{S}|$ denotes the cardinality of a set $\mathcal{S}$.

II. Problem Formulation

A. *Traffic Network Model*

We adopt the traffic network model with sinusoidal approximation proposed in [6]. In what follows, we will describe the model and explain this sinusoidal approximation technique.

A traffic network is described by a directed graph $G = (\mathcal{S} \cup \{\epsilon\}, \mathcal{L})$. Each node in $\mathcal{S} = \{1, 2, \ldots, |S|\}$ represents a signalized intersection and ϵ is the dummy intersection (source) for traffic originating outside the network. Let $N = |S|+1$ be the number of intersections including the dummy intersection. For notational simplicity, the dummy node ϵ is also referred to as node N. Each directed edge in $\mathcal{L}$ represents a traffic link between two intersections/signals and the vehicle queue associated with the link. For each $l \in \mathcal{L}$, $\tau(l) \in \mathcal{S}$ indicates its upstream intersection and $\sigma(l) \in \mathcal{S}$ represents the downstream intersection which serves the queue of the link. $\mathcal{E} = \{l \in \mathcal{L}, \tau(l) = \epsilon\} \subset \mathcal{L}$ is the set of entry links that direct exogenous traffic from the dummy intersection (source) to the network; other links are non-entry links and the travel time from its upstream to downstream intersections is denoted by λ_l.

The vehicle queue associated with each link $l \in \mathcal{L}$ has length $q_l(t)$ at time t. A continuous-time fluid queue model is considered so that $q_l(t)$ evolves as

$$\dot{q}_l(t) = a_l(t) - d_l(t) \tag{1}$$

where $a_l(t)$ is the arrival rate for vehicles arriving from the upstream intersection and $d_l(t)$ is the departure rate that depends on the downstream signal actuation. Both $a_l(t)$ and $d_l(t)$ are in units of vehicles per hour. All intersections are operated under fixed time control [18]. This means that the links at each intersection are activated (served by the intersection) for a fixed amount of time according to a periodic signal activation sequence of non-conflicting links. The period of the sequence is the cycle time of the intersection, and all intersections are assumed to have a common cycle time $T = 1$ time unit.

The signal offset $\theta_s \in [0, 1)$ for an intersection $s \in \mathcal{S}$ represents a phase difference of the signal activation sequence from a global clock. The offsets are the design variables of the optimization problem. For each link $l \in \mathcal{L}$, its queue is served by $\sigma(l)$ at times $n + \theta_{\sigma(l)} + \gamma_l$ for $n = 0, 1, 2, \ldots$, where $\gamma_l \in [0, 1)$ is the link's green split that represents the time difference of the midpoint of the activation time for the link and the beginning of the offset time $\theta_{\sigma(l)}$. For $l, k \in \mathcal{L}$, the turn ratio β_{lk} denotes the fraction of vehicles that are routed to link k upon exiting link l. Note that $\sum_{k\in\mathcal{L}} \beta_{lk} \leq 1$ and $\beta_{lk} = 0$ when $\sigma(l) \neq \tau(k)$.

Similarly to [6], assume that the network is in the periodic steady state and approximate all arrivals, departures, and queue lengths by sinusoid functions with period $T = 1$. Specifically, the departure rate of link l is assumed to be

$$d_l(t) = f_l(1 + \cos(2\pi(t - \theta_{\sigma(l)} - \gamma_l))) \tag{2}$$

where f_l is the average departure rate of link l. By defining $z_j = e^{i2\pi\theta_j}$ for $j \in \mathcal{S}$ and $D_l = f_l e^{-i2\pi\gamma_l}$, one can write the departure rate at link l as

$$d_l(t) = f_l + \mathrm{Re}\left(e^{i2\pi t} D_l \bar{z}_{\sigma(l)}\right). \tag{3}$$

Since vehicles arrive at a non-entry link from its upstream links after a delay equal to the travel time, the arrival rate of a non-entry link $l \in \mathcal{L} \setminus \mathcal{E}$ is given by

$$a_l(t) = \sum_{k\in\mathcal{L}} \beta_{kl} d_k(t - \lambda_l). \tag{4}$$

The periodic steady state assumption implies that the average arrival rate is the same as the average departure rate at each link [18], i.e., $\int_0^1 a_l(t)dt = \int_0^1 d_l(t)dt$. Therefore, we have $\sum_{k\in\mathcal{L}} \beta_{kl} f_k = f_l$. Then, the arrival rate can be further expressed as

$$a_l(t) = f_l + \text{Re}\left(e^{i2\pi t} A_l \bar{z}_{\tau(l)}\right) \tag{5}$$

where $A_l = e^{-i2\pi\lambda_l} \sum_{k\in\mathcal{L}} \beta_{kl} D_k$.

For an entry link $l \in \mathcal{E}$, the approximation assumes that

$$a_l(t) = f_l + \alpha_l \cos(2\pi(t-\phi_l))) = f_l + \text{Re}\left(e^{i2\pi t} A_l \bar{z}_{\tau(l)}\right) \tag{6}$$

where $z_{\tau(l)} = e^{i2\pi\theta_N} = 1$ with the offset θ_N of the dummy intersection ϵ (intersection N) defined to be 0 in the above equation, $\alpha_l \le f_l$ is the relative amplitude of the arrival peak minus the average rate, $A_l = \alpha_l e^{-2\pi\phi_l}$, and $\phi_l \in [0,1)$ is the offset for the center of the arrival peak.

It follows from the queue dynamics (1), departure rate (3) and arrival rate (5)-(6) of the links that the queue length $q_l(t)$ of each link $l \in \mathcal{L}$ evolves according to the equation

$$\dot{q}_l(t) = a_l(t) - d_l(t) = \text{Re}\left(e^{i2\pi t}(A_l\bar{z}_{\tau(l)} - D_l\bar{z}_{\sigma(l)})\right).$$

Accordingly, the average queue length at link l, denoted by Q_l, is given by

$$Q_l = \frac{1}{2\pi}|(A_l\bar{z}_{\tau(l)} - D_l\bar{z}_{\sigma(l)})|.$$

B. Offset Optimization Problem

The average queue lengths Q_l where $l \in \mathcal{L}$, are important performance metrics for traffic networks. Following the approach in [6], we formulate the offset optimization problem as selecting offsets $\theta_s, s = 1,2,\ldots,N$ with the goal of minimizing the total average squared queue length. Note that the queue lengths are invariant to a constant shift for all θ_s where $s = 1,2,\ldots,N$. Therefore, instead of restricting $\theta_N = 0$ for the dummy intersection ϵ, one can allow θ_N be a variable that takes any value in the interval $[0,1]$ and set the offset of each intersection $s \in \mathcal{S}$ to be the relative offset $\theta_s - \theta_N$. Then, the offset optimization problem can be formulated as follows:

$$\begin{aligned} \underset{\theta_1,\ldots,\theta_N}{\text{minimize}} \quad & \sum_{l\in\mathcal{L}} Q_l^2 && (7)\\ \text{subject to} \quad & Q_l = \frac{1}{2\pi}|(A_l\bar{z}_{\tau(l)} - D_l\bar{z}_{\sigma(l)})| \\ & z_s = e^{i2\pi\theta_s}, \quad s = 1,2,\ldots,N. \end{aligned}$$

Note that the queue length of each link satisfies

$$\begin{aligned} Q_l^2 =& \frac{1}{(2\pi)^2}|(A_l\bar{z}_{\tau(l)} - D_l\bar{z}_{\sigma(l)})|^2 \\ =& \frac{1}{(2\pi)^2}(|A_l|+|D_l|)^2 \\ &- \frac{1}{(2\pi)^2}(2|A_l||D_l|+\bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)}). \end{aligned}$$

Since $(|A_l|+|D_l|)^2$ is constant, minimizing $\sum_{l\in\mathcal{L}} Q_l^2$ is equivalent to maximizing

$$\begin{aligned} &\sum_{l\in\mathcal{L}}(2|A_l||D_l|+\bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)}) \\ &= \sum_{l\in\mathcal{L}}(|A_l||D_l||z_{\tau(l)}|^2+|A_l||D_l||z_{\sigma(l)}|^2 \\ &\qquad + \bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)}) \\ &= z^H W z \end{aligned} \tag{8}$$

where $z \in \mathbb{C}^N$ is the variable vector with the entries $z[j] = z_j$, and $W \in \mathbb{C}^{N\times N}$ is a Hermitian matrix whose elements are given by:

$$W[j,j] = \sum_{l\in\mathcal{L}:\tau(l)=j} |A_l||D_l| + \sum_{l\in\mathcal{L}:\sigma(l)=j} 2|A_l||D_l| \tag{9}$$

$$W[j,k] = \frac{1}{2}\Big(\sum_{l\in\mathcal{L}:\tau(l)=j,\sigma(l)=k} \bar{D}_l A_l + \sum_{l\in\mathcal{L}:\tau(l)=k,\sigma(l)=j} D_l \bar{A}_l\Big) \quad \text{for } j \ne k. \tag{10}$$

Lemma 1. *The matrix W is positive semidefinite.*

Proof. For every $z \in \mathbb{C}^N$, it follows from (8) that

$$\begin{aligned} z^H W z = &\sum_{l\in\mathcal{L}}(|A_l||D_l||z_{\tau(l)}|^2+|A_l||D_l||z_{\sigma(l)}|^2 \\ &+ \bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)}). \end{aligned}$$

In addition, for every link l it holds that

$$\begin{aligned} &\bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)} \\ &= 2\text{Re}(\bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)}) \ge -2|A_l||D_l||z_{\tau(l)}||z_{\sigma(l)}|. \end{aligned}$$

Therefore,

$$\begin{aligned} z^H W z \ge &\sum_{l\in\mathcal{L}}(|A_l||D_l||z_{\tau(l)}|^2+|A_l||D_l||z_{\sigma(l)}|^2 \\ &\qquad - 2|A_l||D_l||z_{\tau(l)}||z_{\sigma(l)}| \\ = &\sum_{l\in\mathcal{L}}|A_l||D_l|(|z_{\tau(l)}|+|z_{\sigma(l)}|)^2 \ge 0. \end{aligned}$$

This concludes that W is positive semidefinite. □

Now, one can formulate the offset optimization problem (7) as the following QCQP:

$$\begin{aligned} \underset{z\in\mathbb{C}^N}{\text{maximize}} \quad & z^H W z \\ \text{subject to} \quad & |z[j]|^2 = 1, \quad j = 1,2,\ldots,N. \end{aligned} \tag{11}$$

Given a solution $\hat{z}$ to the QCQP (11), one can obtain the optimal offsets of the traffic network via the equation

$$\theta_s = \frac{1}{2\pi}(\angle\hat{z}[s] - \angle\hat{z}[N]) \tag{12}$$

for every intersection $s \in \mathcal{S}$.

Remark 1. *Note that the QCQP* (11) *formulated in this paper is subtly different from the one considered in [6]. Specifically, the W matrix in* (11) *is positive semidefinite, which will enable us to compute the approximation ratio of the relaxation.*

III. Low-complexity Optimization Algorithm

In the previous section, the offset optimization problem was cast as the QCQP (11) that maximizes a convex objective function subject to nonconvex constraints. This formulation is aligned with the classic MAX-CUT problem in combinatorial optimization [9]. Indeed, the MAX-CUT problem is recovered by forcing each z_j to be real (then they are binary). The QCQP (11) is NP-hard from its similarity to the MAX-CUT problem. Unless P=NP, we should focus the attention on finding a ρ-approximation algorithm; that is, a polynomial-time algorithm that generates a solution of value at least a factor ρ times the globally optimal value.

This section describes a new technique based on the celebrated Goemans–Williamson algorithm [10] for MAX-CUT that solves (11) with a performance guarantee of $\pi/4 \geq 0.785$ (i.e., it is a $\pi/4$-approximation algorithm). Moreover, the complexity of the proposed algorithm is near-linear, requiring at most $O(\omega^{6.5}N^{1.5})$ time and $O(\omega^4 N)$ memory to generate an approximate solution. Here, ω is the maximum clique size that is a graph-theoretic parameter associated with the underlying traffic network. We will discuss more about ω later in the section.

In practice, the proposed algorithm performs even better than its provable guarantees. Our numerical results in Section IV find that the algorithm has a linear empirical time complexity, and that every solution enjoys a performance guarantee of more than 0.99.

A. $\pi/4$-approximation Algorithm

Inspired by the Goemans–Williamson algorithm in [10], one can interpret (11) as an optimization problem over the 1-dimensional unit ball. This means that the problem restricts each decision variable $z[j] \in \mathbb{C}$ to be a 1-dimensional complex vector of unit norm. Replacing each $z[j]$ by an N-dimensional unit vector v_j leads to the relaxation:

$$\begin{aligned}\underset{v_1,\ldots,v_N\in\mathbb{C}^N}{\text{maximize}} \quad & \sum_{j=1}^{N}\sum_{k=1}^{N} W[j,k] v_j^H v_k \\ \text{subject to} \quad & \|v_j\|^2 = 1, \quad j = 1,\ldots,n.\end{aligned} \tag{13}$$

This nonconvex problem can be reformulated into a convex problem by a change of variables $X = [v_j^H v_k] \in \mathbb{C}^{N\times N}$:

$$\begin{aligned}\underset{X\in\mathbb{C}^{N\times N}}{\text{maximize}} \quad & \operatorname{tr}(WX) \\ \text{subject to} \quad & X[j,j] = 1, \quad j = 1,\ldots,N, \\ & X \succeq 0.\end{aligned} \tag{14}$$

Problem (14) is an SDP for which an interior-point method is able to compute an optimal solution in polynomial time. We can recover a corresponding globally-optimal set of vectors $\hat{v}_1,\ldots,\hat{v}_N \in \mathbb{C}^N$ for (13) by factoring $\hat{X} = \hat{V}^H\hat{V}$ and taking each $\hat{v}_j$ to be the j-th column of the matrix $\hat{V}$.

Following Goemans and Williamson, one can project an optimal set of vectors $\hat{v}_1,\ldots,\hat{v}_N \in \mathbb{C}^N$ for (13) back onto $\mathbb{C}$ by randomized rounding

$$\hat{z}_j = r^H\hat{v}_j / |r^H\hat{v}_j|. \tag{15}$$

Here, $r \in \mathbb{C}^N$ is a random vector whose real and imaginary parts are selected independently and identically from the N-dimensional Gaussian distribution, as in

$$r = r_1 + ir_2 \qquad r_1, r_2 \sim \mathcal{N}(0, I) \tag{16}$$

where $\mathcal{N}(0, I)$ denotes the N-dimensional Gaussian distribution with identity covariance matrix and zero mean.

This rounding method can be repeated with several choices of r, and we select the candidate solution with the best objective value. The follow result states that this randomization rounding offers a remarkably high-quality solution.

Theorem 1. *Given the optimal solution $\hat{v}_1,\ldots,\hat{v}_N \in \mathbb{C}^N$ for (13), define the candidate solution $\hat{z} \in C^N$ for* (11) *using (15) for each $\hat{z}_j \in \mathbb{C}$, in which $r \in \mathbb{C}^N$ is selected as in (16). Then,*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} W[j,k]\hat{v}_j^H\hat{v}_k \geq opt_{QCQP} \geq \mathbb{E}\left[\hat{z}^H W \hat{z}\right] \geq \frac{\pi}{4} opt_{QCQP},$$

where opt_{QCQP} is the globally optimal value of (11).

Proof. The first bound is true because (13) is a relaxation of (11), and the second bound holds because $\hat{z}_1,\ldots,\hat{z}_N \in \mathbb{C}$ is not necessarily optimal for (11). The third bound follows from a result of [19], noting that $W \succeq 0$ from Lemma 1. ☐

Remark 2. *It is worth noting that when the solution $\hat{X}$ to the SDP (14) is* rank-one *(i.e., if there exists $x \in \mathbb{C}^N$ such that $\hat{X} = xx^H$), the relaxation (13) becomes exact and the first two bounds in Theorem 1 become equalities. This special situation occurs for certain types of networks [20] and the offsets obtained from the SDP solution achieves optimal performance for these cases [6]. In general, however, the matrix $\hat{X}$ has a rank strictly greater than one. Nevertheless, we observe in numerical experiments in Section IV that the associated performance guarantee (i.e. the ratio between the upper- and lower-bounds on the performance) exceeds* 99% *for every case.*

In summary, this subsection describes a $\pi/4$-approximation algorithm for the QCQP (11) of the offset optimization problem that comprises three key steps:

1) Solve the SDP relaxation (14) and obtain the optimal solution $\hat{X} \in \mathbb{C}^{N\times N}$;
2) Recover the N optimal N-dimensional vectors $\hat{v}_1,\ldots,\hat{v}_N \in \mathbb{C}^N$ for (14); and
3) Round $\hat{v}_1,\ldots,\hat{v}_N \in \mathbb{C}^N$ into $\hat{z}_1,\ldots,\hat{z}_N \in \mathbb{C}$ using the randomized procedure in (15).

Standard algorithms implement these three steps with a combined complexity of $O(N^{4.5})$ time and $O(N^2)$ memory, with the first step dominating the overall complexity. These figures are polynomial, and hence "efficient" in theory. In practice, however, they become prohibitive for large-scale traffic networks with more than 1000 intersections.

B. Efficient Implementation for Large-but-sparse Networks

When the traffic network is large but sparse in the sense that it has a *bounded treewidth* [17], the $\pi/4$-approximation algorithm described in the previous subsection can be implemented in near-linear $O(N^{1.5})$ time and linear $O(N)$ memory.

Algebraically, given a *fill-reducing permutation* matrix P, we factor the matrix W into a Cholesky factor L satisfying

$$LL^H = PWP^H, \quad L \text{ is lower-triangular}, \quad L[j,j] \geq 0. \tag{17}$$

Let $\mathcal{I}_1, \ldots, \mathcal{I}_N \subseteq \{1, \ldots, N\}$ be the column index sets from the sparsity pattern of L defined by

$$\mathcal{I}_j = \{k \in \{1, \ldots, N\} : L[k,j] \neq 0\}. \tag{18}$$

From the column index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$, define a set of parent pointers $p : \{1, \ldots, n\} \to \{1, \ldots, n\}$ for the associated *elimination tree*:

$$p(j) = \begin{cases} j & |\mathcal{I}_j| = 1, \\ \min_i\{i > j : i \in \mathcal{I}_j\} & |\mathcal{I}_j| > 1. \end{cases} \tag{19}$$

The column index sets $\mathcal{I}_1, \ldots, \mathcal{I}_n$ and the elimination tree constitute a *tree decomposition* for the network. Denote the maximum clique size for the tree decomposition as

$$\omega = \max_j \quad |\mathcal{I}_j|. \tag{20}$$

Then, a network is said to have a bounded treewidth if we are able to find a fill-reducing permutation P such that $\omega = O(1)$, or equivalently, the Cholesky factor L contains at most $O(N)$ nonzero elements.

Note that ω is a graph-theoretic parameter that only depends on the sparsity pattern of the matrix W. From the definition (10) of W, the entry $W[j,k]$ is zero if no link connects between the j-th intersection and the k-th intersection. Therefore, real-world traffic networks are likely to satisfy the bounded treewidth property because the associated graphs are *planar* and *sparse*. We will discuss the idea of treewidth and the tree decomposition in details in the following subsection.

Now, suppose that the network has a bounded treewidth and we are given a fill-reducing permutation P, its associated index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$ with $\omega = O(1)$, and the parent pointers p. Without loss of generality, we may assume that $P = I$; otherwise, we can solve the permuted problem with $\tilde{W} = PWP^T$, and reverse the ordering $z = P^T\tilde{z}$ once a solution $\tilde{z}$ has been computed. The remainder of this subsection describes an implementation of the $\pi/4$-approximation algorithm that achieves the claimed $O(N^{1.5})$ time and $O(N)$ memory complexity. Our presentation uses only linear algebra, and can be efficiently realized via function calls to standard numerical linear algebra libraries.

We use the clique tree conversion technique of [11] to reformulate (14) into a reduced-complexity problem with the variables $X_j \in \mathbb{C}^{|\mathcal{I}_j| \times |\mathcal{I}_j|}, j = 1, \ldots, N$:

$$\begin{aligned} \underset{X_1,\ldots,X_N}{\text{minimize}} \quad & \sum_{j=1}^{N} \operatorname{tr}(W_j X_j) \\ \text{subject to} \quad & X_j[k,k] = 1, \quad j = 1, \ldots, N, \quad k = 1, \ldots, |\mathcal{I}_j| \\ & R_{p(j),j}(X_j) = R_{j,p(j)}(X_{p(j)}), \\ & X_j \succeq 0, \quad j = 1, \ldots, N, \end{aligned} \tag{21}$$

whose solution $\hat{X}_j$ is related to the solution $\hat{X}$ of (14) by

$$\hat{X}[\mathcal{I}_j, \mathcal{I}_j] = \hat{X}_j, \quad j = 1, \ldots, n.$$

The matrices $W_1, \ldots, W_j$ in (21) are defined to satisfy

$$\sum_{j=1}^{N} \operatorname{tr}(W_j X[\mathcal{I}_j, \mathcal{I}_j]) = \operatorname{tr}(WX)$$

with respect to the original W matrix, over all Hermitian choices of $X = X^H \in \mathbb{C}^{N \times N}$. The linear operator $R_{k,j} : \mathbb{C}^{|\mathcal{I}_j| \times |\mathcal{I}_j|} \to \mathbb{C}^{|\mathcal{I}_k| \times |\mathcal{I}_k|}$ is defined to output the overlapping elements of two principal submatrices indexed by $\mathcal{I}_k$ and $\mathcal{I}_j$, given the latter as the argument:

$$R_{k,j}(X[\mathcal{I}_j, \mathcal{I}_j]) = X[\mathcal{I}_k \cap \mathcal{I}_j, \mathcal{I}_k \cap \mathcal{I}_j] = R_{j,k}(X[\mathcal{I}_k, \mathcal{I}_k]). \tag{22}$$

The associated constraints $R_{p(j),j}(X_j) = R_{j,p(j)}(X_{p(j)})$ in (21) are known as the *overlap constraints*.

With the assumption that $|\mathcal{I}_j| \leq \omega = O(1)$, an interior-point method solves the converted problem (21) to ϵ-accuracy in $O(N^{3.5} \log \epsilon^{-1})$ time and $O(N^2)$ memory. In the worst case, the converted problem is only marginally easier to solve than the original problem (14), which has time complexity $O(N^{4.5} \log \epsilon^{-1})$ and memory complexity $O(N^2)$.

A recent result of [16] shows that the complexity guarantees for (21) can be significantly improved to near-linear $O(N^{1.5} \log \epsilon^{-1})$ time and linear $O(N)$ memory complexities by a simple *dualization* procedure (note that [16] studies real-valued SDPs, but the results can be generalized in a straightforward way to complex-valued SDPs). To explain, we begin by putting (21) into primal canonical form:

$$\begin{aligned} \underset{x_1,\ldots,x_N \in \mathbb{C}^{N^2}}{\text{minimize}} \quad & \sum_{j=1}^{N} w_j^T x_j \\ \text{subject to} \quad & \begin{bmatrix} R_1 & \cdots & R_N \\ M_1 & & 0 \\ & \ddots & \\ 0 & & M_N \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}, \\ & x_j \in \mathcal{K}_j, \quad j = 1, \ldots, N. \end{aligned} \tag{23}$$

Each variable $x_j = \operatorname{vec}(X_j)$ (respectively, $w_j = \operatorname{vec}(W_j)$) is the vectorization of X_j (respectively, W_j) and each $\mathcal{K}_j$ is the corresponding positive semidefinite cone. The matrices $R_1, \ldots, R_N$ implement the overlap constraints in (22). Each constraint matrix M_j isolates the diagonal of X_j, as in $M_j \operatorname{vec}(X_j)[k] = X_j[k,k]$. Next, we construct the dualized problem, posed in dual canonical form:

$$\begin{aligned} \underset{y_1,\ldots,y_N}{\text{maximize}} \quad & -\sum_{j=1}^{N} w_j^T y_j \\ \text{subject to} \quad & \begin{bmatrix} R_1 & \cdots & R_N \\ M_1 & & 0 \\ & \ddots & \\ 0 & & M_N \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} + s_0 = \begin{bmatrix} 0 \\ \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}, \\ & -y_j + s_j = 0, \quad j = 1, \ldots, N \\ & s_0 \in \{0\}^{N+1}, \quad s_j \in \mathcal{K}_j. \end{aligned} \tag{24}$$

Here, $\{0\}^{N+1}$ denotes the so-called "equality-constraint cone", whose dual cone is a free variable of dimension $N+1$. We proceed to solve the dualized problem (24) using a

Algorithm 1 Positive semidefinite matrix completion

Input. The column index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$ defined in (18) and the solutions $\hat{X}_1, \ldots, \hat{X}_j$ to (21).

Output. The solution $\hat{X}$ to (14) in the form of $\hat{X} = F^{-H} D F^{-1}$, where D is a diagonal matrix and F is a lower-triangular matrix with the same sparsity pattern as L.

Algorithm. Iterate over $j \in \{1, \ldots, N\}$ in any order. Set $F[j,j] = 1$ and solve for the j-th column of D and F by finding any $D[j,j]$ and $F[\mathcal{I}_j \backslash \{j\}, j]$ that satisfy

$$\hat{X}_j \begin{bmatrix} 1 \\ F[\mathcal{I}_j \backslash \{j\}, j] \end{bmatrix} = \begin{bmatrix} D[j,j] \\ 0 \end{bmatrix}.$$

general-purpose interior-point method like SeDuMi, SDPT3, and MOSEK.

Theorem 2. *A general-purpose interior-point method solves the SDP* (21) *by solving its dual canonical form* (24) *to ϵ-accuracy in*

$$O(\omega^{6.5} N^{1.5} \log \epsilon^{-1}) \text{ time and } O(\omega^4 N) \text{ memory.}$$

Proof. This theorem is a complex-valued version of the results developed in [16] for real-valued SDPs. The details are omitted for brevity, but a sketch of the proof will be provided below. Loosely, a general-purpose interior-point method solves an order-θ linear conic program posed in the canonical form to ϵ-accuracy in $O(\sqrt{\theta} \log \epsilon^{-1})$ iterations. The cone in (24) has order $\theta = O(\omega N)$, so the interior-point method converges in $O(\omega^{0.5} n^{0.5} \log \epsilon^{-1})$ iterations. At each iteration, the complexity is dominated by the Newton subproblem whose Newton matrix can be partitioned into N blocks of size $O(\omega^2)$, and the associated block-sparsity pattern coincides with the adjacency matrix of the elimination tree. Such matrices can be factored with zero block-fill using a minimum degree ordering in $O(\omega^6 N)$ time and $O(\omega^4 N)$ memory. □

Finally, we use Algorithm 1 (adopted from [21]) to recover the solution $\hat{X}$ of (14) given solutions $\hat{X}_1, \ldots, \hat{X}_j$ to (21). The algorithm implicitly recovers the dense matrix $\hat{X}$ in a sparse factored form $\hat{X} = F^{-H} D F^{-1}$, where D is diagonal and F is lower-triangular with the same sparsity pattern as L in (17). As such, the factorization allows the randomized rounding described earlier in (15) to be efficiently performed

$$F^H s = D^{1/2} r, \qquad z_j = s_j / |s_j|, \tag{25}$$

by solving a sparse triangular system of equations.

In summary, this subsection presents a reduced-complexity implementation of a $\pi/4$-approximation algorithm for the QCQP (11) of the offset optimization problem given a fill-reducing permutation matrix P with $\omega = O(1)$. The full algorithm is described as follows:

1) Permute the data as $W \leftarrow P W P^H$, compute the Cholesky factor L as in (17), and determine the index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$ and the elimination tree parent vector p, as in (18) and (19).
2) Use the clique tree conversion technique of [11] to reformulate (14) into (21).
3) Vectorize (21) into (23) and use the dualization technique of [21] to reformulate (23) into (24).
4) Solve (24) as a dual canonical problem using a general-purpose interior-point method.
5) Recover the solution $\hat{X}$ of (14) in the sparse factored form $\hat{X} = F^{-H} D F^{-1}$ using Algorithm 1.
6) Recover a choice of $\hat{z}_1, \ldots, \hat{z}_N \in \mathbb{C}$ via the randomized rounding method (25) and reverse the fill-reducing permutation $\hat{z} \leftarrow P^H \hat{z}$.

It is shown below that Step 4 of the algorithm dominates the the overall complexity.

Corollary 3. *Given a fill-reducing permutation P, a choice of $\hat{z}_1, \ldots, \hat{z}_N \in \mathbb{C}$ that satisfy the bounds in Theorem 1 can be computed with the same time and memory complexity as quoted in Theorem 2.*

Proof. Step 1 is dominated by the Cholesky factorization step, for $O(\omega^3 N)$ time and $O(\omega^2 N)$ memory. Steps 2 and 3 are algebraic manipulations, requiring $O(\omega^2 N)$ time and memory. Step 4 uses $O(\omega^{6.5} N^{1.5} \log \epsilon^{-1})$ time and $O(\omega^4 N)$ memory according to Theorem 2. Step 5 is dominated by solving N linear systems of up to size $\omega \times \omega$ for $O(\omega^3 N)$ time and $O(\omega^2 N)$ memory. Finally, Step 6 can be performed by back-substitution in $O(\omega N)$ time and memory. Clearly, the complexity of the entire procedure is dominated by Step 4. □

C. Fill-reducing Permutation

The previous subsection describes an efficient implementation of the $\pi/4$-approximation algorithm for the offset optimization problem. Its key ingredient is a fill-reducing permutation P for a large-but-sparse network. More specifically, we need to search for a P that yields a small ω. Fill-reducing permutations are closely associated with the concept of tree decompositions in graph theory.

Definition 4. A *tree decomposition* of a graph $G = (\mathcal{S} \cup \{\epsilon\}, \mathcal{L})$ of is a pair $(\mathcal{I}, T)$, where T is a tree of N vertices and $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$ contains subsets of $\mathcal{S} \cup \{\epsilon\}$, one for each node of T, such that:

1) (Node cover) For every node $s \in \mathcal{S} \cup \{\epsilon\}$, there exists $\mathcal{I}_j \in \mathcal{I}$ such that $s \in \mathcal{I}_j$;
2) (Edge cover) For every link $l \in \mathcal{L}$, there exists $\mathcal{I}_k \in \mathcal{I}$ such that $\sigma(l) \in \mathcal{I}_k$ and $\tau(l) \in \mathcal{I}_k$; and
3) (Running intersection) If $s \in \mathcal{I}_j$ and $s \in \mathcal{I}_k$, then we also have $s \in \mathcal{I}_m$ for every $\mathcal{I}_m$ that lies on the path from $\mathcal{I}_j$ to $\mathcal{I}_k$ in the tree T.

Definition 5 ([17])**.** The *width* of a tree decomposition $(\mathcal{I}, T)$ is $\omega - 1$ where $\omega = \max\{|\mathcal{I}_1|, \ldots, |\mathcal{I}_N|\}$ as defined in (20), i.e., one less than the maximum number of elements in any subset $\mathcal{I}_k \in \mathcal{I}$. The *treewidth* of a network is the minimum width amongst all tree decompositions. The network is said to have a *bounded treewidth* if its treewidth is $O(1)$, i.e., independent of the number of nodes N.

From the definition, a fill-reducing permutation P with $\omega = O(1)$ exists if and only if the network has a bounded treewidth. The problem of computing an optimal P that minimizes ω

coincides with the problem of computing the exact treewidth of the network, which is well-known to be NP-complete [22]. For bounded treewidth networks known *a priori* to have small $\omega \ll N$, the optimal P can be computed in $O(2^{\omega}N)$ time [23], which is linear to N but exponential to ω. In practice, it is much easier to compute a "good-enough" P with a small but suboptimal value of ω, using one of the heuristics originally developed for the fill-reduction problem in numerical linear algebra.

In the case of traffic networks, the graphs are almost *planar* by construction, because the vast majority of roads do not cross without intersecting. Planar graphs with N nodes have treewidths of at most $O(\sqrt{N})$, attained by grid graphs; a tree decomposition within a constant factor of the optimal can be explicitly computed using the planar separator theorem and a nested dissection ordering. Practical traffic networks tend to have treewidths possibly much smaller than the $O(\sqrt{N})$ figure. While local networks tend to resemble grids, inter-area networks interconnecting wider regions are more tree-like. Accordingly, their treewidth is usually bounded by the square-root of the size of the largest grid, which is fairly small even for networks typically thought of as "grid-like" such as Manhattan and Downtown Los Angeles.

Our numerical examples in Section IV use the simple approximate minimum degree algorithm in generating a choice of P. This approximately coincides with the simple "greedy algorithm", and does not typically enjoy strong guarantees. Regardless, the algorithm is extremely fast, generating permutations for graphs containing millions of nodes and edges in a matter of seconds.

IV. Numerical Experiments

To test the developed method, we generate test cases using real-world traffic networks from the OpenStreet Map data [24]. For each test case, we consider a rectangular area of the real-world map. We construct a traffic network by assuming that all intersections in the area are signalized, and a link is added from one intersection to another one if there is a road/way between the two intersections following the corresponding direction.

For each network, the green splits γ_l's are set based on the orientations of the links. In particular, the green split of a link at an intersection is the angle of the link from the longitude line at the intersection. Travel times λ_l's are assigned to be proportional to the lengths of the links. The turn ratios β_{lk}'s are set to be such that the traffic traveling straight is twice the traffic making each turn direction. The average flow f_l's of all entry links are the same constant and the flows of non-entry links are calculated from the turn ratios by solving $f_l = \sum_{k\in\mathcal{L}} \beta_{kl} f_k$ for all $l \in \mathcal{L}$.

The first set of the networks is generated using the map of the Berkeley area as shown in Fig. 1. The Berkeley-1 network has 405 intersections and 1122 links connecting the intersection while the Berkeley-4 has 7000 and 12176 intersections that includes the network of Berkeley, Oakland, and their surrounding areas. The second set of networks is generated from the map of the Manhattan area as in Fig. 2, and the third set of networks is based on the Downtown Los Angeles area as in Fig. 3.

The numerical results are presented in Table I. The lower bound (LB) on the squared queue length is obtained from $\sum_{j=1}^{N} \mathrm{tr}(W_j X_j)$ of the solution of the reduced-complexity SDP (21) in Step 4 of the algorithm. The upper bound (UB) is the result from the best solution $\hat{z}$ in 200 runs of the randomized rounding method in Step 6 of the algorithm. The algorithm is implemented in a Matlab code, and the numerical experiments are performed on an HP SE1102 server with 2 quad-core 2.5GHz Xeon and 24 GB memory.

As observed in Table I, the gap between the upper and lower bounds is very small for all cases, and therefore the proposed algorithm is able to solve the SDP relaxations and compute near-globally optimal solutions for networks with up to twelve thousand intersections. In terms of time complexity, Fig. 4 shows that the runtime scales almost linearly with respect to the number of intersections in the network. This agrees with the claim of Theorem 2. Note that the parameter ω of the networks is bounded by 50. This observation supports the argument that real-world traffic networks indeed have a desirable sparsity structure so that the treewidths are much smaller than the square root bound $O(\sqrt{N})$ in practice.

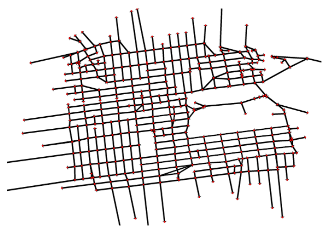

Fig. 1. Berkeley Network

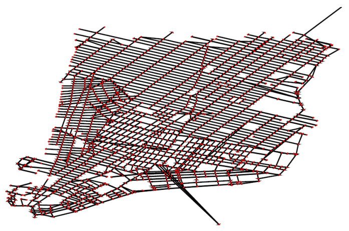

Fig. 2. Manhattan Network

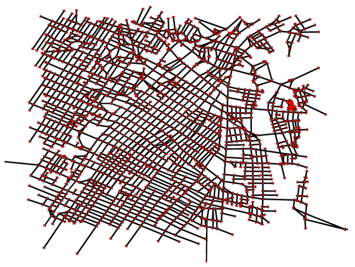

Fig. 3. Los Angeles Network

TABLE I
NETWORK CASES

Cases	$\|\mathcal{S}\|$	$\|\mathcal{L}\|$	ω	LB	UB	Runtime (s)
Berkeley-1	405	1122	14	79209	79447	22
Berkeley-2	2036	5789	35	477449	479736	2745
Berkeley-3	7000	19222	41	1588518	1597013	12838
Berkeley-4	12176	33725	42	2795242	2811025	20942
Manhattan-1	1432	2745	31	301366	302965	2005
Manhattan-2	2016	3854	31	417186	419601	1516
Manhattan-3	3923	7841	37	780878	787402	4932
Manhattan-4	9968	20945	39	2022526	2040154	29117
Los Angeles-1	733	2180	22	182811	183440	1305
Los Angeles-2	1838	5170	36	458209	460905	1305
Los Angeles-3	3062	8838	43	747804	752099	7045
Los Angeles-4	4239	12773	50	1139072	1147166	20733

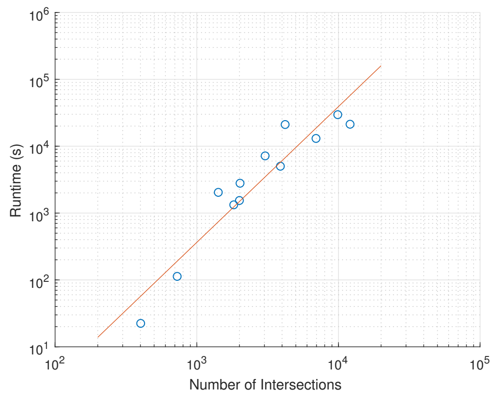

Fig. 4. Runtime Versus Number of Intersections

V. CONCLUSION

We adopt a recently proposed QCQP formulation for the traffic signal offset optimization problem. By exploiting sparsity structures of traffic networks, we propose an algorithm based on tree decompositions to optimally solve an SDP relaxation of the offset optimization problem. Given the optimal solution to the SDP relaxation, we further generate near-optimal offsets using a randomized rounding method. The overall algorithm has the time complexity $O(N^{1.5})$ and memory complexity $O(N)$ for networks with bounded treewidths. We show in various real-world traffic networks that the treewidths are clearly bounded. Numerical experiments verify the underlying complexity result, and the algorithm is able to obtain almost optimal solutions for networks with up to twelve thousand intersections.

REFERENCES

[1] R. E. Allsop, "Selection of offsets to minimize delay to traffic in a network controlled by fixed-time signals," *Transportation Science*, vol. 2, no. 1, pp. 1–13, 1968.

[2] N. H. Gartner and J. D. Little, "The generalized combination method for area traffic control," 1973.

[3] J. D. Little, M. D. Kelson, and N. H. Gartner, "Maxband: A versatile program for setting signals on arteries and triangular networks," 1981.

[4] N. H. Gartner and C. Stamatiadis, "Arterial-based control of traffic flow in urban grid networks," *Mathematical and computer modelling*, vol. 35, no. 5-6, pp. 657–671, 2002.

[5] N. H. Gartner and C. Stamatiadis, "Progression optimization featuring arterial-and route-based priority signal networks," *Journal of Intelligent Transportation Systems*, vol. 8, no. 2, pp. 77–86, 2004.

[6] S. Coogan, E. Kim, G. Gomes, M. Arcak, and P. Varaiya, "Offset optimization in signalized traffic networks via semidefinite relaxation," *Transportation Research Part B: Methodological*, vol. 100, pp. 82–92, 2017.

[7] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, 2010.

[8] E. S. Kim, C.-J. Wu, R. Horowitz, and M. Arcak, "Offset optimization of signalized intersections via the burer-monteiro method," in *American Control Conference (ACC), 2017*, pp. 3554–3559, IEEE, 2017.

[9] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, pp. 85–103, Springer, 1972.

[10] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[11] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion i: General framework," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 647–674, 2001.

[12] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 199–211, 2015.

[13] R. Madani, M. Ashraphijuo, and J. Lavaei, "Promises of conic relaxation for contingency-constrained optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1297–1307, 2016.

[14] Y. Zhang, R. Madani, and J. Lavaei, "Conic relaxations for power system state estimation with line measurements," *IEEE Transactions on Control of Network Systems*, 2017.

[15] R. G. Cowell, P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media, 2006.

[16] R. Y. Zhang and J. Lavaei, "Sparse semidefinite programs with near-linear time complexity," *arXiv preprint arXiv:1710.03475*, 2017.

[17] N. Robertson and P. D. Seymour, "Graph minors. ii. algorithmic aspects of tree-width," *Journal of algorithms*, vol. 7, no. 3, pp. 309–322, 1986.

[18] A. Muralidharan, R. Pedarsani, and P. Varaiya, "Analysis of fixed-time control," *Transportation Research Part B: Methodological*, vol. 73, pp. 81–90, 2015.

[19] A. M.-C. So, J. Zhang, and Y. Ye, "On approximating complex quadratic optimization problems via semidefinite programming relaxations," *Mathematical Programming*, vol. 110, no. 1, pp. 93–110, 2007.

[20] S. Sojoudi and J. Lavaei, "Exactness of semidefinite relaxations for nonlinear optimization problems with underlying graph structure," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 1746–1778, 2014.

[21] L. Vandenberghe, M. S. Andersen, *et al.*, "Chordal graphs and semidefinite optimization," *Foundations and Trends® in Optimization*, vol. 1, no. 4, pp. 241–433, 2015.

[22] S. Arnborg, D. G. Corneil, and A. Proskurowski, "Complexity of finding embeddings in ak-tree," *SIAM Journal on Algebraic Discrete Methods*, vol. 8, no. 2, pp. 277–284, 1987.

[23] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks, "Approximating treewidth, pathwidth, frontsize, and shortest elimination tree," *Journal of Algorithms*, vol. 18, no. 2, pp. 238–255, 1995.

[24] "Openstreetmap." https://wiki.openstreetmap.org.