Large-Scale Traffic Signal Offset Optimization

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

***Abstract*—We consider the traffic signal offset optimization problem that coordinates the offsets of signalized intersections to reduce vehicle queues in large-scale 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 the tree decomposition and a dualization procedure to reformulate the large-scale problem into a reduced-complexity complex-valued 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.**

***Index Terms*—Transportation networks, traffic signal control, optimization, semidefinite programming.**

I. Introduction

Transportation has played a crucial role in human societies. In a transportation network, most intersections are regulated by traffic signals to ensure the safety and mobility of roadway users. The operation of urban transportation heavily depends on the control of these signalized intersections. The primary signal control parameters are cycle lengths, splits, offsets and phasing sequences. Among these parameters, offset is the key component to achieve coordination for traffic in a network. The offset is the phase difference between cycles of intersections. For a link/road in a traffic network, the offset of the two ends of the link/road determines how the two intersection cycles align with each other. Along an arterial, an appropriate selection of offsets can align the signals to create a "green wave" for vehicles to drive without stop. For a general traffic network, the control of signal offsets can reduce vehicle idling time at red lights, hence decreases traffic queues and delays in the entire network. Offset optimization seeks to determine traffic 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. Given the link delay functions, the offset optimization problem is formulated as minimizing the sum of all delays of the traffic network. Under some restrictions on the link delay functions or on the network topology, methods based on dynamic programming [2], [3] and mixed-integer programming formulations [4–6] can be used to solve the offset optimization problem. In addition to certain restrictions, these methods generally suffer from 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. One main challenge of this approach is the modeling of traffic flows and vehicle queues. A recent work [7] 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 [8]. For smaller networks where the SDP relaxation is exact (i.e., the SDP relaxation has the same optimal objective value as the QCQP), [7] 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 [7] 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 [9] 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 SDPs. 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 [7] for the offset optimization problem. The QCQP can be transformed into a complex-valued quadratic maximization problem. This formulation is very similar to the classic MAX-CUT problem in combinatorial optimization [10]. Inspired by the Goemans–Williamson algorithm for MAX-CUT [11], we

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 grant N00014-17-1-2933, DARPA grant D16AP00002, AFOSR grant FA9550- 17-1-0163, ARO grant W911NF-17-1-0555, and NSF Awards 1545116 and 1807260. Preliminary version of this paper will appear in the Proceedings of the 57th IEEE Conference on Decision and Control, 2018 [1].

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 SDPs are 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 possesses 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 [12]. Similar methods have shown success in solving large-scale power networks [13–16] and in machine learning [17]. Using a tree decomposition, we first reformulate the SDP relaxation into a reduced-complexity SDP. We then adopt a dualization technique recently developed in [18] 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. The entire algorithm is implemented in the complex number domain, which significantly improves on the runtime of the results of our conference version [1].

When a network is "sparse" in the sense that it has a bounded treewidth [19], 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 verify that the proposed algorithm has a linear empirical time complexity and it can compute near-optimal offsets for networks with up to twelve thousand intersections within an hour. 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 $x \in \mathbb{C}$, $\mathrm{Re}(x)$, $\mathrm{Im}(x)$, $\bar{x}$, $\angle x$, 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 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 $\mathcal{I}_j, \mathcal{I}_k \subseteq \{1,2,\ldots,n\}$, 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

To determine traffic signal offsets, we adopt the traffic network model with sinusoidal approximation proposed in [7]. In what follows, we will first describe the model and explain this sinusoidal approximation technique. Then, using this model, we formulate a mathematical optimization problem to select offsets that minimize the lengths of vehicle queues of the networks.

A. Traffic Network Model

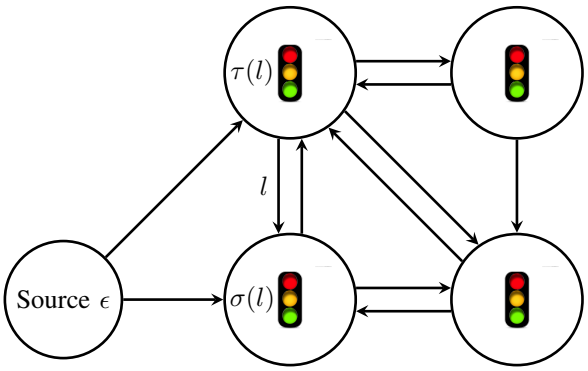

Fig. 1. Traffic Network

Consider a traffic network described by a directed graph $G = (\mathcal{S} \cup \{\epsilon\}, \mathcal{L})$. Each node of the graph represents an intersection; node $i \in \mathcal{S} = \{1,2,\ldots,|S|\}$ represents a signalized intersection and node 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. 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. There is no need to explicitly model links that exit the network because exiting traffic are considered in the calculation of turn ratios, which will be defined later.

The vehicle queue associated with each link $l \in \mathcal{L}$ has length $q_l(t)$ at time t. The queue length $q_l(t)$ follows a continuous-time fluid queue model given by

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

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 intersection signal. Both $a_l(t)$ and $d_l(t)$ are in units of vehicles per hour.

Vehicles coming from a link are allowed to pass through an intersection when the link is activated by the traffic signal, i.e., green light for the link. To avoid collision, each signal switches among activation patterns of non-conflicting links according to a signal control sequence. All intersections are assumed to operated under fixed time control [20] with common cycle. This means that the signal control sequence of each intersection has a fixed periodic cycle, and all intersections 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 the phase difference of the signal control sequence from a global clock. For each link $l \in \mathcal{L}$, vehicles from its queue is allowed to pass through intersection $\sigma(l)$ at times $n + \theta_{\sigma(l)} + \gamma_l$ for $n = 0, 1, 2, \ldots$, where $\gamma_l \in [0, 1)$ is called 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 $\beta_{lk} \in [0, 1]$ denotes the fraction of vehicles that are routed to link k upon exiting link l. When $\sigma(l) \neq \tau(k)$, $\beta_{lk} = 0$ because the two links are not connected. For every link $l \in \mathcal{L}$ it holds that

$$\sum_{k\in\mathcal{L}} \beta_{lk} \leq 1$$

where strict inequality in the above equation models the situation that a fraction of vehicles exit the network via an unmodeled link from intersection $\sigma(l)$.

Similarly to [7], we 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

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

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

$$d_l(t) = f_l + \text{Re}\left(e^{i2\pi t} D_l \bar{z}_{\sigma(l)}\right). \tag{2}$$

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

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

The periodic steady-state assumption implies that the average arrival rate is the same as the average departure rate at each link [20], 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

$$a_l(t) = f_l + \text{Re}\left(e^{i2\pi t} A_l \bar{z}_{\tau(l)}\right) \tag{3}$$

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

$$a_l(t) = f_l + \alpha_l \cos(2\pi(t - \phi_l))) = f_l + \text{Re}\left(e^{i2\pi t} A_l \bar{z}_{\tau(l)}\right) \tag{4}$$

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, $\alpha_l \leq f_l$ 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 (2) and arrival rate (3)-(4) 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 [7], 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 to 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:

$$\begin{aligned} \underset{\theta_1,\ldots,\theta_n}{\text{minimize}} \ & \sum_{l\in\mathcal{L}} Q_l^2 \\ \text{subject to } & Q_l = \frac{1}{2\pi} |(A_l \bar{z}_{\tau(l)} - D_l \bar{z}_{\sigma(l)})| \\ & z_s = e^{i2\pi\theta_s}, \quad s = 1, 2, \ldots, n. \end{aligned} \tag{5}$$

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

$$\begin{aligned} & \sum_{l\in\mathcal{L}} (2|A_l||D_l| + \bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)}) \\ &= \sum_{l\in\mathcal{L}} (|A_l||D_l||z_{\tau(l)}|^2 + |A_l||D_l||z_{\sigma(l)}|^2 \\ &\quad + \bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)}) \\ &= z^H W z \end{aligned} \tag{6}$$

where $z \in \mathbb{C}^n$ is the vector of variables z_j, and $W \in \mathbb{C}^{n\times n}$ is a Hermitian matrix whose elements are given by:

$$W_{j,j} = \sum_{l\in\mathcal{L}:\tau(l)=j} |A_l||D_l| + \sum_{l\in\mathcal{L}:\sigma(l)=j} 2|A_l||D_l| \tag{7}$$

$$W_{j,k} = \frac{1}{2}\Big(\sum_{l\in\mathcal{L}:\tau(l)=j,\sigma(l)=k} \bar{D}_l A_l + \sum_{l\in\mathcal{L}:\tau(l)=k,\sigma(l)=j} D_l \bar{A}_l \Big) \quad \text{for } j \neq k. \tag{8}$$

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

Proof. For every $z \in \mathbb{C}^n$, it follows from (6) that

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

In addition, for every link l it holds that

$$\bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)} + D_l \bar{A}_l z_{\tau(l)} \bar{z}_{\sigma(l)} = 2\mathrm{Re}(\bar{D}_l A_l \bar{z}_{\tau(l)} z_{\sigma(l)}) \geq -2|A_l||D_l||z_{\tau(l)}||z_{\sigma(l)}|.$$

Therefore,

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

This concludes that W is positive semidefinite. □

Now, one can formulate the offset optimization problem (5) as the following QCQP:

$$\begin{aligned} &\underset{z\in\mathbb{C}^n}{\text{maximize}} \; z^H W z \\ &\text{subject to } |z_j|^2 = 1, \quad j = 1, 2, \ldots, n. \end{aligned} \tag{9}$$

Given a solution $\hat{z}$ to the QCQP (9), one can obtain the optimal offsets of the traffic network via the equation

$$\theta_s = \frac{1}{2\pi}(\angle \hat{z}_s - \angle \hat{z}_n) \tag{10}$$

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

Remark 1. *Note that the QCQP* (9) *formulated in this paper is subtly different from the one considered in [7]. Specifically, the diagonal elements of the matrix W in [7] are all zero so the matrix is not positive semidefinite. In our formulation, the matrix W in* (9) *is positive semidefinite, which will enable us to compute the approximation ratio of the relaxation.*

III. Approximation Algorithm

In the previous section, offset optimization was cast as the optimization problem (9) that maximizes a convex objective function subject to nonconvex constraints. This QCQP formulation results in a nonconvex optimization problem. In fact, such nonconvex QCQP is known to be NP-hard [21]. Unless P=NP, we have to focus on finding an efficient approximation algorithm with polynomial complexities for large-scale traffic networks.

Note that this formulation of offset optimization has a similar structure as the QCQP formulation of the classic MAX-CUT problem in combinatorial optimization [10]. Indeed, when the variables z_j in problem (9) are forced to be real numbers, each z_j is either 1 or -1 and the problem becomes an instance of MAX-CUT. In other words, the QCQP (9) can be viewed as a complex version of the MAX-CUT problem.

Based on the celebrated Goemans–Williamson algorithm [11] for MAX-CUT, we provide below a polynomial complexity algorithm that solves (9) with a performance guarantee of $\pi/4 \geq 0.785$ (i.e., the value of the solution is at least a factor $\pi/4$ times the globally optimal value). In practice, the proposed algorithm might perform even better than the provable guarantees. Our numerical results in Section V find that every solution enjoys a performance guarantee of more than 0.99.

Following the idea of the Goemans–Williamson algorithm, one can interpret (9) as an optimization problem over the one-dimensional unit sphere. This means that the problem restricts each decision variable $z_j \in \mathbb{C}$ to be an one-dimensional unit vector. Replacing each one-dimensional vector $z_j \in \mathbb{C}$ by an n-dimensional unit vector $v_j \in \mathbb{C}^n$ leads to the relaxation:

$$\begin{aligned} &\underset{v_1,\ldots,v_n\in\mathbb{C}^n}{\text{maximize}} \; \sum_{j=1}^n \sum_{k=1}^n W_{j,k} v_j^H v_k \\ &\text{subject to } \|v_j\|^2 = 1, \quad j = 1, \ldots, n. \end{aligned} \tag{11}$$

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

$$\begin{aligned} &\underset{X\in\mathbb{C}^{n\times n}}{\text{maximize}} \; \mathrm{tr}(WX) \\ &\text{subject to } X_{j,j} = 1, \quad j = 1, \ldots, n, \\ &\qquad\qquad X \succeq 0. \end{aligned} \tag{12}$$

Lemma 2. *Problem* (12) *is a relaxation of* (9)*, and therefore, its value gives an upper-bound for the optimal value of* (9)*.*

Proof. Given any feasible solution $z \in \mathbb{C}^n$ of (9), let $v_j = (z_j, 0, 0, \ldots, 0) \in \mathbb{C}^n$ for $j = 1, 2, \ldots, n$. Then, $v_j^H v_k = \bar{z}_j z_k$ for all $j, k = 1, 2, \ldots, n$. Consequently, $(v_1, v_2, \ldots, v_n)$ is feasible for (12) and its objective value in (12) is the same as the objective value of z in (9). □

Problem (12) is an SDP for which an interior-point method is able to compute an optimal solution in polynomial time with a given accuracy. We can recover a corresponding globally-optimal set of vectors $\hat{v}_1, \ldots, \hat{v}_N \in \mathbb{C}^n$ for (11) by factoring $\hat{X} = \hat{V}^H \hat{V}$ and taking each $\hat{v}_j$ to be the j-th column of the matrix $\hat{V}$.

Remark 2. *The SDP* (12) *can also be generated from* (9) *using a standard SDP relaxation procedure [8]. Specifically, by adding a rank constraint rank$(X) = 1$ in* (12)*, one obtain the original QCQP* (9) *because any rank-one matrix X can be factored into $X = zz^H$. The relaxation* (12) *becomes exact if its solution $\hat{X}$ has rank one. This special situation occurs for certain types of networks [22] and the offsets obtained from the SDP solution achieves optimal performance for these cases [7]. In general, however, the solution $\hat{X}$ of* (12) *has a*

rank strictly greater than one. Nevertheless, we observe in our numerical experiments in Section V that the associated performance guarantee (i.e. the ratio between the upper- and lower-bounds on the performance) exceeds 99% *for every case.*

In spirit of the Goemans–Williamson idea method, one can project an optimal set of vectors $\hat{v}_1, \ldots, \hat{v}_n \in \mathbb{C}^n$ for (11) back onto the one-dimensional unit sphere in $\mathbb{C}$ by randomized rounding

$$s_j = r^H \hat{v}_j, \qquad \hat{z}_j = s_j/|s_j|. \tag{13}$$

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

$$r = r_1 + ir_2, \qquad r_1, r_2 \sim \mathcal{N}(0, I) \tag{14}$$

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 (11), define the candidate solution $\hat{z} \in C^n$ for* (9) *using (13) for each $\hat{z}_j \in \mathbb{C}$, in which $r \in \mathbb{C}^n$ is selected as in (14). 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 (9) *and $\mathbb{E}[\cdot]$ is the expectation operator.*

Proof. The first bound is true because (11) is a relaxation of (9) by Lemma 2, and the second bound holds because $\hat{z}_1, \ldots, \hat{z}_N \in \mathbb{C}$ is not necessarily optimal for (9). The third bound follows from a result of [21], noting that $W \succeq 0$ from Lemma 1. □

In summary, this section describes a $\pi/4$-approximation algorithm for the QCQP (9) of the offset optimization problem that comprises two key steps:

1) Solve the SDP relaxation (12) and obtain the optimal solution $\hat{X} \in \mathbb{C}^{n\times n}$; and
2) 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 (13).

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.

IV. Efficient Implementation for Sparse Networks

When a traffic network is large but sparse in the sense that it has a *bounded treewidth* [19], we show in this section that the approximation algorithm described in the previous section can be implemented in near-linear $O(n^{1.5})$ time and linear $O(n)$ memory.

In the following, we first describe the concept of tree decomposition and use it to convert the original problem to a reduced-complexity problem. Then, we further simplify the complexity to obtain a near-linear time approximation algorithm for offset optimization.

A. Tree Decomposition

For a traffic network $G = (\mathcal{S} \cup \{\epsilon\}, \mathcal{L})$, the graph theoretical concepts of tree decomposition and treewidth are defined as follows:

Definition 1. *A* tree decomposition *of a graph G of is a pair $(\mathcal{I}, T)$, where $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ are n subsets of nodes of G, and T is a tree with vertices $\mathcal{I}$, such that:*

1) *(Node cover) For every node s of G, there exists $\mathcal{I}_j \in \mathcal{I}$ such that $s \in \mathcal{I}_j$;*
2) *(Edge cover) For every edge l of G, 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 2 ([19])**.** *The* width *of a tree decomposition $(\mathcal{I}, T)$ is $\omega - 1$ where*

$$\omega = \max_j \quad |\mathcal{I}_j|, \tag{15}$$

i.e., the width is 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, the empty graph has treewidth of zero, and tree and forest graphs have treewidths of one. Basically, the treewidth of a graph indicates how tree-like the graph is. The treewidth can be viewed as a sparsity criterion which determines the complexities of may problems related to a graph. The problem of computing the exact treewidth of a graph is known to be NP-complete [23]. For bounded treewidth networks known *a priori* to have small $\omega \ll n$, the treewidth and the corresponding tree decomposition can be determined in $O(2^\omega n)$ time [24]. In practice, it is much easier to compute a "good-enough" tree decomposition 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 our implementation, we use the simple approximate minimum degree algorithm in generating a tree decomposition [25]. 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.

Algebraically, a tree decomposition of our traffic network can also be described by a *fill-reducing permutation* matrix P. More specifically, given a permutation matrix $P \in \mathbb{R}^{n\times n}$, we can factor the matrix W of the network into a Cholesky factor L satisfying

$$LL^H = PWP^H, \quad L \text{ is lower-triangular}, \quad L_{j,j} \geq 0. \tag{16}$$

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

$$\mathcal{I}_j = \{k \in \{1, \ldots, n\} : L_{k,j} \neq 0\}. \tag{17}$$

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

$$p(j) = \begin{cases} j & |\mathcal{I}_j| = 1, \\ \min_i\{i > j : i \in \mathcal{I}_j\} & |\mathcal{I}_j| > 1. \end{cases} \tag{18}$$

Lemma 3. *The collection of the column index sets $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ together with the tree T constructed by nodes $\mathcal{I}$ and edges $\{(\mathcal{I}_j, \mathcal{I}_{p(j)}), j = 1, 2, \ldots, n\}$ constitute a tree decomposition for the network G.*

Proof. According to [26], the pair $(\mathcal{I}, T)$ forms a tree decomposition of W. From the definition (8) of W, the entry $W_{j,k}$ is zero if no link connects between the j-th intersection and the k-th intersection. Therefore, the sparsity pattern of the matrix W is the same as the traffic network G. □

For networks with a bounded treewidth, we are able to find a tree decomposition whose width is $\omega = \max_j |\mathcal{I}_j| = O(1)$. Since the Cholesky factor L of W has at most ω nonzero elements per column, L of such networks will be a sparse matrix containing at most $O(n)$ nonzero elements.

In the case of real-world 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 may 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 relatively small even for networks typically thought of as "grid-like" such as Manhattan and Downtown Los Angeles.

B. Clique Tree Conversion and Recovery

Using the concept of tree decomposition, this subsection describe the clique tree conversion technique of [12] to simplify the $\pi/4-$approximation algorithm proposed in the previous section.

Suppose that the network has a bounded treewidth and we are given a tree decomposition with $\omega = O(1)$ represented by a fill-reducing permutation P, its associated index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$, and the parent pointers p. From now on, without loss of generality, we 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.

Given the tree decomposition, the clique tree conversion technique reformulates (12) into a reduced-complexity problem with the variables $X_j \in \mathbb{C}^{|\mathcal{I}_j| \times |\mathcal{I}_j|}, j = 1, \ldots, n$:

$$\begin{aligned} \underset{X_1, \ldots, X_n}{\text{minimize}} \ & \sum_{j=1}^{n} \operatorname{tr}(W_j X_j) \\ \text{subject to } & (X_j)_{k,k} = 1, \quad j = 1, \ldots, n, \quad k = 1, \ldots, |\mathcal{I}_j| \\ & R_{p(j),j}(X_j) = R_{j,p(j)}(X_{p(j)}), \\ & X_j \succeq 0, \quad j = 1, \ldots, n, \end{aligned} \tag{19}$$

where $W_1, \ldots, W_j$ are matrices satisfying

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

with respect to the original W matrix, over all Hermitian choices of $X \in \mathbb{C}^{n \times n}$. The exact method to construct $W_1, \ldots, W_j$ can be found in [18]. 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:

$$R_{k,j}(X_{\mathcal{I}_j, \mathcal{I}_j}) = X_{\mathcal{I}_k \cap \mathcal{I}_j, \mathcal{I}_k \cap \mathcal{I}_j} = R_{j,k}(X_{\mathcal{I}_k, \mathcal{I}_k}). \tag{20}$$

The associated constraints $R_{p(j),j}(X_j) = R_{j,p(j)}(X_{p(j)})$ in (19) are known as the *overlap constraints*.

From the bounded treewidth property, this conversion reduces the number of decision variables from $O(n^2)$ for X in (12) to $O(\omega n)$ for $\{X_j, j = 1, \ldots, n\}$ in (19).

Lemma 4. *The solutions $\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n$ of* (19) *are related to the solution $\hat{X}$ of* (12) *by*

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

Proof. The proof is omitted as its essentially the same as the real-valued version in [12]. □

The above relation allows us to recover the solution $\hat{X}$ of (12) from solutions $\hat{X}_1, \ldots, \hat{X}_j$ of (19). Note that $\hat{X}$ is generally a dense matrix, so simply forming the matrix would push the overall complexity up to quadratic $O(n^2)$ time and memory. Fortunately, the Cholesky factorization of $\hat{X}$ is sparse due to the bounded treewidth property. Therefore, we compute $\hat{X}$ implicitly in factorized form 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 (16). This can be done by the following Algorithm 1 adopted from [26].

Algorithm 1 Positive semidefinite matrix completion

Input. The column index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$ defined in (17) and the solutions $\hat{X}_1, \ldots, \hat{X}_j$ to (19).

Output. The solution $\hat{X}$ to (12) 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 $F_{\mathcal{I}_j \setminus \{j\}, j}$ that satisfy

$$\hat{X}_j \begin{bmatrix} 1 \\ F_{\mathcal{I}_j \setminus \{j\}, j} \end{bmatrix} = \begin{bmatrix} D_{j,j} \\ 0 \end{bmatrix}.$$

With the factorized solution $\hat{X} = F^{-H}DF^{-1}$, we can now efficiently implement the randomized rounding procedure described earlier in (13). Specifically, from $\hat{X} = F^{-H}DF^{-1} = \hat{V}^H\hat{V}$ we obtain $\hat{V} = D^{1/2}F^{-1}$. Then, (13) is equivalent to

$$F^H s = D^{1/2} r, \qquad \hat{z}_j = s_j/|s_j|. \tag{21}$$

Since F is a lower-triangular matrix (with the same sparsity pattern as L), one can compute $\hat{z}$ from (21) by solving a sparse triangular system of equations in $O(\omega n)$ time.

In summary, this subsection presents a reduced-complexity implementation of a $\pi/4$-approximation algorithm for the QCQP (9) of the offset optimization problem given a tree decomposition with $\omega = O(1)$. The main steps are described as follows:

1) Reformulate (12) into the reduced complexity problem (19).
2) Solve (19) to obtain solutions $\hat{X}_1, \ldots, \hat{X}_n$.
3) Recover the solution of (12) in the sparse factored form $\hat{X} = F^{-H}DF^{-1}$ using Algorithm 1.
4) Recover a choice of $\hat{z}_1, \ldots, \hat{z}_N \in \mathbb{C}$ via the randomized rounding method (21).

We will show later that the complexity of the overall algorithm is dominated by Step 3, i.e., solving the semidefinite problem (19). When an interior-point method is used to solve (19), the complexity will be mainly determined by the solution of a normal equation in each interior-point iteration. Even though the problem has a sparsity structure, the normal equation is general dense and it results in $O(n^{3.5})$ time and $O(n^2)$ memory complexities. We show next that the complexities of solving (19) can be reduced by exploiting the sparsity structure using a dualization procedure.

C. Dualization

A recent result of [18] shows that the complexity of solving the real-valued version of (19) can be significantly improved to near-linear $O(n^{1.5})$ time and linear $O(n)$ memory complexities by a *dualization* procedure. We present in this subsection a complex-valued version of the algorithm of [18] for the traffic offset optimization problem.

To solve (19), we begin by putting (19) into primal canonical form:

$$\begin{aligned} \underset{x_1,\ldots,x_n \in \mathbb{C}^{n^2}}{\text{minimize}} \ & \sum_{j=1}^{n} \bar{w}_j^H x_j \\ \text{subject to} \ & \begin{bmatrix} N_{11} & \cdots & N_{1n} \\ & \ddots & \\ N_{n1} & \cdots & N_{nn} \\ M_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}, \\ & x_j \in \mathcal{K}_j, \quad j = 1, \ldots, n. \end{aligned} \tag{22}$$

Each variable $x_j = \mathrm{vec}(X_j)$ (respectively, $w_j = \mathrm{vec}(W_j)$) is the vectorization of X_j (respectively, W_j) and each $\mathcal{K}_j$ is the corresponding positive semidefinite cone. The matrices N_{jk} implement the overlap constraints in (20). That is, for each j, the j-th block row $N_{j1}, \ldots, N_{jn}$ implements the overlap constraint between $\mathcal{I}_j$ and its parent $\mathcal{I}_p(j)$. Therefore, the j-th block row has at most two nonzero sub-blocks: $N_{jk} = 0$ except $k = j$ or $k = p(j)$. Each constraint matrix M_j isolates the diagonal of X_j, as in $(M_j x_j)_k = (X_j)_{k,k}$.

Let N and M denote the matrices for the constraints:

$$N = \begin{bmatrix} N_{11} & \cdots & N_{1n} \\ & \ddots & \\ N_{n1} & \cdots & N_{nn} \end{bmatrix}, \quad M = \begin{bmatrix} M_1 & & 0 \\ & \ddots & \\ 0 & & M_n \end{bmatrix}.$$

Then, the complexity of each step of the interior-point iteration solving (22) depends on the sparsity pattern of $\tilde{M}\tilde{M}^H$ where $\tilde{M} = [N^H, M^H]^H$. Despite the nice sparsity structure of $\tilde{M}$, the matrix $\tilde{M}\tilde{M}^H$ is generally dense (see [18] for an example). Therefore, it takes $O(n^{3.5})$ time and $O(n^2)$ memory to solve (22) using an interior-point solver.

On the other hand, the matrix $\tilde{M}^H\tilde{M}$ is sparse from the block sparsity structure of N and M.

Lemma 5. *The matrix $\tilde{M}^H\tilde{M}$ has $O(\omega^4 n)$ nonzero elements, and it takes $O(\omega^6 n)$ operations to compute $\tilde{M}^H\tilde{M}$ from N and M.*

Proof. This is a corollary of the result of [18]. In particular, M is the adjacency matrix of an empty graph, so the block sparsity structure of $\tilde{M}^H\tilde{M}$ is the same as the sparsity of the adjacency matrix of the tree T of the tree decomposition. Then, $\tilde{M}^H\tilde{M}$ has $O(n)$ nonzero blocks, and each of the blocks has at most $O(\omega^4)$ nonzero elements. The computation of $\tilde{M}^H\tilde{M}$ is done by adding up $O(\omega^2 n)$ sets of blocks with $O(\omega^4)$ elements which takes $O(\omega^6 n)$ operations. □

In order to exploit the sparsity structure of $\tilde{M}^H\tilde{M}$, one way is to *dualize* the problem [26]. The dualized problem of (22), posted in dual canonical form, is given by:

$$\begin{aligned} \underset{y_1,\ldots,y_n}{\text{maximize}} \ & -\sum_{j=1}^{N} \bar{w}_j^H y_j \\ \text{subject to} \ & M \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} + s_0 = \begin{bmatrix} 0 \\ \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}, \\ & -y_j + s_j = 0, \quad j = 1, \ldots, n \\ & s_0 \in \{0\}^{N+1}, \quad s_j \in \mathcal{K}_j. \end{aligned} \tag{23}$$

Here, $\{0\}^{N+1}$ denotes the so-called "equality-constraint cone", whose dual cone is a free variable of dimension $n+1$.

Since (23) is the dual problem, with a general-purpose interior-point method like SeDuMi, SDPT3, and MOSEK, each iteration involves solving a normal equation of matrix $\tilde{M}\tilde{M}^H$. We then achieve the desired complexity results from the sparsity of $\tilde{M}\tilde{M}^H$.

Theorem 2. *A general-purpose interior-point method solves the SDP* (19) *by solving its dual canonical form* (23) *to ϵ-accuracy in*

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

Proof. The proof in [18] for real-valued SDPs can be adopted to prove this theorem. First, note that 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 (23) has order $\theta = O(\omega n)$ from the construction of the tree decomposition, so the interior-point method converges in $O(\omega^{0.5}n^{0.5}\log\epsilon^{-1})$ iterations.

At each interior-point iteration, the complexity is dominated by the solution of the normal equations that are linear equations described by a matrix H whose sparsity pattern is the same as $\tilde{M}^H\tilde{M}$. From Lemma 5, forming $\tilde{M}^H\tilde{M}$ requires $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory. We then have the stated memory complexity, and the time complexity result is obtained by multiplying $O(\omega^{0.5}n^{0.5}\log\epsilon^{-1})$ with $O(\omega^6 n)$. □

D. Overall Algorithm

This section presents a reduced-complexity implementation of a $\pi/4$-approximation algorithm for the QCQP (9) of the offset optimization problem. The full algorithm is described as follows:

1) Compute a tree decomposition for the traffic network G and its fill-reducing permutation P using the minimum degree algorithm.
2) Permute W as $W \leftarrow PWP^H$, compute the Cholesky factor L as in (16), and determine the index sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$ and the parent pointers p, as in (17) and (18).
3) Use the clique tree conversion technique to reformulate (12) into (19).
4) Convert (19) to the dualized problem (23).
5) Solve (23) as a dual canonical problem using a general-purpose interior-point method to obtain solutions $\hat{X}_1, \ldots, \hat{X}_n$ of (19).
6) Recover the solution of (12) in the sparse factored form $\hat{X} = F^{-H}DF^{-1}$ using Algorithm 1.
7) Recover a choice of $\hat{z}_1, \ldots, \hat{z}_n \in \mathbb{C}$ via the randomized rounding method (21). This randomization step can be run several times to obtain a solution with the best objective value.
8) Reverse the fill-reducing permutation $\hat{z} \leftarrow P^H\hat{z}$.

Corollary 3. *The proposed algorithm generates a choice of $\hat{z}_1, \ldots, \hat{z}_N \in \mathbb{C}$ that satisfy the bounds in Theorem 1 and can be computed with the same time and memory complexity as described in Theorem 2.*

Proof. The minimum degree algorithm in Step 1 takes $O(\omega n)$ time and memory. Step 2 is dominated by the Cholesky factorization step, for $O(\omega^3 n)$ time and $O(\omega^2 n)$ memory. Steps 3 and 4 are algebraic manipulations, requiring $O(\omega^2 n)$ time and memory. Step 5 uses $O(\omega^{6.5}n^{1.5}\log\epsilon^{-1})$ time and $O(\omega^4 n)$ memory according to Theorem 2. Algorithm 1 in Step 6 is dominated by solving n linear systems of up to size ω^2 for $O(\omega^3 n)$ time and $O(\omega^2 n)$ memory. The rounding method of (21) in Step 7 can be performed by back-substitution in $O(\omega n)$ time and memory. Finally, Step 8 takes $O(n)$ time and memory to obtain an approximate solution with the guarantees in Theorem 1. □

Remark 3. *The offset optimization problem* (9) *is formulated as a complex-valued QCQP. This complex-valued QCQP has an equivalent real-valued formulation. Specifically, consider $z = x - iy$ where $x, y \in \mathbb{R}^n$ are the real and imaginary parts of z. Then,* (9) *is equivalent to*

$$\underset{x,y\in\mathbb{R}^n}{\text{maximize}} \; [x^H y^H] \begin{bmatrix} \mathit{Re}\,(W) & \mathit{Im}\,(W) \\ -\mathit{Im}\,(W) & \mathit{Re}\,(W) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (24)$$
$$\text{subject to } x_j^2 + y_j^2 = 1, \quad j = 1, 2, \ldots, n.$$

One can then follow a similar procedure to solve this transformed real-valued problem as in our conference version [1]. This way, approximately optimal offsets can also be obtained in the same near-linear time complexity as the algorithm proposed in this section. However, transforming offset optimization into the real-valued problem has a main drawback: the sparsity structure of the real-valued problem (24) *depends on the matrix* $\begin{bmatrix} \mathit{Re}\,(W) & \mathit{Im}\,(W) \\ -\mathit{Im}\,(W) & \mathit{Re}\,(W) \end{bmatrix}$ *whose treewidth is twice the treewidth of the traffic network G. The effect of the doubled treewidth makes this transformation method to be about* 10 *times slower than the proposed algorithm for the networks considered in our numerical experiments. See [27] for such speed-up in optimization solvers using complex numbers instead of real numbers.*

V. Numerical Experiments

To test the developed algorithm for offset optimization, we generate test cases using real-world traffic networks from the OpenStreet Map data [28]. For each test case, we consider a rectangular area of the real-world map. From each area, we construct a traffic network by assuming that all intersections in the area are signalized. Entry links are added for roads/ways entering the target rectangular area, and a non-entry link is added from one intersection to another one if there is a road/way between the two intersections following the corresponding direction. We assume that vehicles travel at a constant speed, so the travel time λ_l of each link is assigned to be proportional to the length of the link in the real-world map. The turn ratios β_{lk}'s are set to be such that, when vehicles entering an intersection form a link, the traffic traveling straight is twice the traffic making each turn direction (left or right). The average flow f_l's of all entry links are assumed to be 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 $l \in \mathcal{L}$.

Since the focus is on the offsets, other signal control parameters are set to be fixed. The cycle lengths of all intersections are the same constant as described in the network model. For each network, the splits and phase sequences are described by the green split parameters γ_l. In the numerical experiments, we do not optimize the green splits γ_l and set them based on the orientations of the links for convenience. In particular, at each intersection, the green split of a link is the angle between the corresponding road/way and the longitude line of the intersection on the real-world map.

The first set of the networks is generated using the map of the Berkeley area as shown in Fig. 2. 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. 3, and the third set of networks is based on the Downtown Los Angeles area as in Fig. 4.

The network parameters and numerical results are presented in Table I. The number of intersections ranges from 405 to 12176 among the networks in our experiments. The lower bound (LB) on the squared queue length is the optimal value of the optimization problem (19) obtained from Step 5 of the algorithm. The optimal value of (19) serves as a bound according to Theorem 1. For each network, 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 performance of the algorithm is much better than the theoretical (worst-case) $\pi/4$ guarantee in Theorem 1. In fact, the gap between the upper and lower bounds on the queue lengths is less than 1% for all cases (99% optimal guarantee). Therefore, despite being an approximation algorithm, the proposed algorithm is able to provide almost globally optimal solutions for the offset optimization problem generated from real-world traffic networks.

In terms of runtime, the algorithm can solve the SDP relaxations and compute near-optimal offsets for networks with up to twelve thousand intersections within an hour. This allows the potential to re-compute offsets every hour based on real-time traffic conditions. Furthermore, Fig. 5 shows that the runtime scales almost linearly with respect to the number of intersections in the network. This agrees with the claim of Corollary 3 and it demonstrates the ability of our algorithm in solving large-scale traffic offset optimization. Note that the parameter ω of the tree decompositions 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.

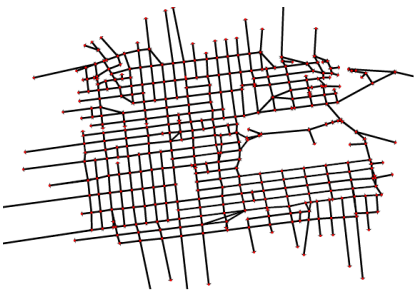

Fig. 2. Berkeley Network

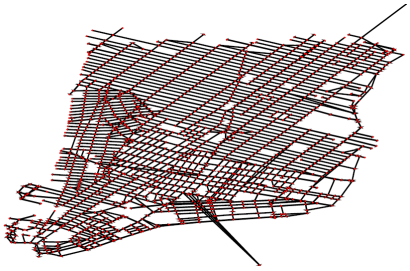

Fig. 3. Manhattan Network

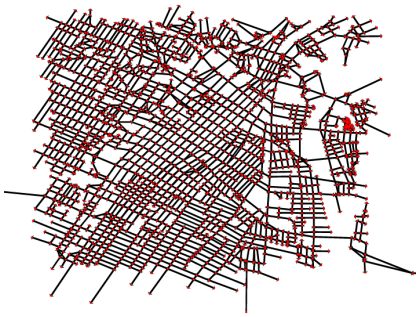

Fig. 4. Los Angeles Network

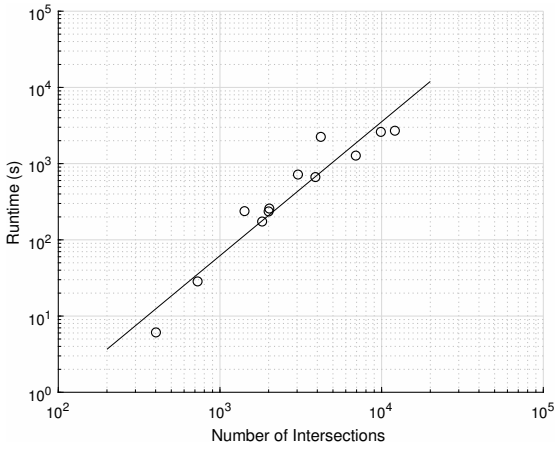

Fig. 5. Runtime Versus Number of Intersections

VI. Conclusion

We adopt a recently proposed QCQP formulation for the traffic signal offset optimization problem. Using the special structure of the QCQP, we propose an approximation algorithm based on a randomized rounding method to generate near-optimal offsets from the optimal solution of an SDP relaxation. By exploiting sparsity structures of traffic networks, we develop an efficient implementation based on the tree decomposition and a dualization procedure to optimally solve the SDP relaxation of the offset optimization problem. The overall algorithm is implemented in the complex-valued domain, which achieves the time complexity of $O(n^{1.5})$ and memory complexity of $O(n)$ for networks with bounded treewidths. We show in various real-world traffic networks that the treewidths are clearly bounded. Numerical experiments

TABLE I
NETWORK CASES

Cases	$\|\mathcal{S}\|$	$\|\mathcal{L}\|$	ω	LB	UB	Runtime (s)
Berkeley-1	405	1122	14	79209	79498	6
Berkeley-2	2036	5789	36	477449	479725	253
Berkeley-3	6979	19222	41	1588518	1597089	1253
Berkeley-4	12176	33725	42	2795240	2810684	2657
Manhattan-1	1430	2748	37	301366	303057	234
Manhattan-2	2016	3854	31	417186	419692	232
Manhattan-3	3923	7841	37	780878	787526	655
Manhattan-4	9968	20945	39	2022529	2039907	2565
Los Angeles-1	733	2180	22	182811	183403	28
Los Angeles-2	1838	5170	36	458209	460708	171
Los Angeles-3	3062	8838	43	747805	752536	707
Los Angeles-4	4239	12773	50	1139072	1146237	2207

verify the underlying complexity result, and the algorithm is able to obtain almost optimal solutions for networks with up to twelve thousand intersections within an hour.

REFERENCES

[1] Y. Ouyang, R. Y. Zhang, J. Lavaei, and P. Varaiya, "Conic approximation with provable guarantee for traffic signal offset optimization," in *IEEE Conference on Decision and Control (CDC), 2018*.

[2] 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.

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

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

[5] 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.

[6] ——, "Progression optimization featuring arterial-and route-based priority signal networks," *Journal of Intelligent Transportation Systems*, vol. 8, no. 2, pp. 77–86, 2004.

[7] 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.

[8] 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.

[9] 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*. IEEE, 2017, pp. 3554–3559.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] R. Madani, S. Sojoudi, G. Fazelnia, and J. Lavaei, "Finding low-rank solutions of sparse linear matrix inequalities using convex optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 725–758, 2017.

[17] 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.

[18] R. Y. Zhang and J. Lavaei, "Sparse semidefinite programs with near-linear time complexity," in *IEEE Conference on Decision and Control (CDC), 2018*.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, 1996.

[26] 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.

[27] J. C. Gilbert and C. Josz, "Plea for a semidefinite optimization solver in complex numbers," Inria Paris, Research Report, Mar. 2017. [Online]. Available: https://hal.inria.fr/hal-01422932

[28] "Openstreetmap," https://wiki.openstreetmap.org.