Large-Scale Traffic Signal Offset Optimization

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

***Abstract*—The offset optimization problem seeks to coordinate and synchronize the timing of traffic signals throughout a network in order to enhance traffic flow and reduce stops and delays. Recently, offset optimization was formulated into a continuous optimization problem without integer variables by modeling traffic flow as sinusoidal. In this paper, we present a novel algorithm to solve this new formulation to near-global optimality on a large-scale. Specifically, we solve a convex relaxation of the nonconvex problem using a tree decomposition reduction, and use randomized rounding to recover a near-global solution. We prove that the algorithm always delivers solutions of expected value at least 0.785 times the globally optimal value. Moreover, assuming that the topology of the traffic network is "tree-like", we prove that the algorithm has near-linear time complexity with respect to the number of intersections. These theoretical guarantees are experimentally validated on the Berkeley, Manhattan, and Los Angeles traffic networks. In our numerical results, the empirical time complexity of the algorithm is linear, and the solutions have objectives within 0.99 times the globally optimal value.**

***Index Terms*—Traffic control, traffic signal timing, offset optimization, convex relaxation, semidefinite programming, tree decomposition**

I. Introduction

In transportation engineering, *traffic signal timing* is the problem of selecting and adjusting the timing of traffic lights in order to reduce congestion and improve traffic flow. This classical problem is commonly formulated as three subproblems:

- *Cycle length optimization*, where the total network is divided into subsections, and a common cycle period is assigned to each subsection;
- *Green split optimization*, where traffic lights within the same intersection are timed to avoid conflicts; and
- *Offset optimization*, where traffic lights over different intersections are coordinated to enhance network-wide performance.

Ideally, these subproblems would be solved simultaneously for the best performance [2], [3]. Owing to issues of computational tractability, however, the established practice is an *iterative* procedure: manually divide the network into subsections, sweep the cycle length over a range of values, and solve the green split and offset optimization subproblems alternatingly for each fixed cycle length [4], [5]. This is precisely the solution procedure implemented in the industry-standard software packages TRANSYT-7F [6, Sec 2.4] and Synchro [7, Ch. 18].

In this paper, we focus our attention on the offset optimization subproblem. The goal is to create *green waves*, in which green lights are synchronized to allow a car to drive through multiple intersections without stopping for a red light, and to maximize the length or *bandwidth* of these green waves. Clearly, green waves are only possible if cycle lengths are the same, or else the synchronization would be lost over time. For this reason, the standard model represents traffic flow as square waves with a common cycle length but separate green times and red times. The exact green splits are assumed to be given and fixed, with the understanding that they will be separately optimized at a later stage.

A. Previous Approaches

The offset optimization problem is highly nonconvex, so solution approaches based on incremental adjustments—such as those implemented in TRANSYT and Sychro—can get stuck at a locally optimal solution. In order to obtain a *globally* optimal solution, the standard approach is to reformulate the problem into a mixed-integer program [8–11] and apply a general-purpose integer programming solver like Gurobi or CPLEX. The latter approach is highly effective on a small scale, but—as is typical for techniques based around integer programming—suffers from severe computational issues as the problem size grow large. In practice, it may not even find a feasible point that does not violate constraints in a reasonable amount time, let alone a globally optimal solution.

Instead, computing globally optimal solutions to large-scale networks generally requires simplifying assumptions. In particular, if a penalty function known as a *link delay function* is assigned to each road link with respect to the offset difference, then dynamic programming can be used to minimize the sum of all link delay functions [4], [12], [13]. For certain network topologies, this approach is guaranteed to compute a globally optimal solution in linear time. However, it is often tricky to choose a link delay function that accurately reflects real-world considerations like queues, delays, and green waves [5], [12]. Also, its use relies on an assumption of link independence that may not be fully realistic [5].

Recently, Coogan et al. [14], [15] proposed an approach that outperforms the link delay function approach described above [15], as well as the incremental adjustment approach found in Synchro [16]. By modeling traffic flow as *sinusoidal*, the problem of minimizing total queue lengths can be posed as a quadratically-constrained quadratic program (QCQP). The QCQP is nonconvex, but can be relaxed into a convex semidefinite program (SDP) using standard techniques, and

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. A preliminary and abridged version has appeared in the Proceedings of the 57th IEEE Conference on Decision and Control, 2018 [1].

solved using an interior-point method. In turn, the solution to the SDP can often recover a globally optimal solution for the QCQP. If desired, the solution can be further refined using TRANSYT or Synchro [17].

Nevertheless, the Coogan et al. [14], [15] approach suffers two serious computational issues that prevent its use on real traffic networks. First, the approach often yields, but does not guarantee, a globally optimal solution. Indeed, such a guarantee is not even possible in general unless P=NP. Moreover, the convex SDP that underpins the approach has a worst-case solution complexity of $O(n^{4.5})$ time and $O(n^2)$ memory. While these figures are formally polynomial, their large exponents limit the number of intersections n to no more than a few hundred.

B. Main Results

Our main contribution in this paper is an algorithm that is *guaranteed* to solve the formulation of Coogan et al. [15] to *near-global* optimality in *near-linear* time. In Section III, we prove that the algorithm always delivers solutions of expected value at least $\pi/4 \geq 0.785$ times the globally optimal value. Moreover, assuming that the topology of the traffic network is "tree-like", we prove in Section IV that the algorithm has near-linear $O(n^{1.5})$ time complexity and linear $O(n)$ memory complexity with respect to the number of intersections n. These theoretical guarantees are experimentally validated in Section V on the Berkeley, Manhattan, and Los Angeles traffic networks. In our numerical results, the algorithm achieves a linear empirical time complexity, and the solutions found all have objectives within 0.99 times the globally optimal value.

Our algorithm works by reformulating offset optimization into a complex-valued quadratically-constrained quadratic program (QCQP) with a similar form to the classic MAX-CUT problem in combinatorial optimization [18], and relaxing the QCQP into a semidefinite program (SDP). Inspired by the Goemans–Williamson algorithm for MAX-CUT [19], we prove that projecting the SDP solution onto a random hyperplane recovers a solution to the QCQP with an approximation ratio of $\pi/4$. We solve the SDP relaxation using the sparsity-exploiting chordal conversion technique of Fukuda et al. [20] and the dualization technique recently developed by Zhang and Lavaei [21]. Directly solving the SDP in the complex domain yields significant improvement on runtime, compared to our previous results in the conference version of this paper [1]. When a network is "sparse" in the sense that it has a bounded treewidth [22], we prove that the overall algorithm has worst-case complexity of $O(n^{1.5})$ time and $O(n)$ memory.

Notation

The sets $\mathbb{R}$ and $\mathbb{C}$ are the real and complex numbers. Subscripts indicate element-wise indexing. The notation $X_{\mathcal{I},\mathcal{J}}$ indicates the submatrix of X indexed by columns sets $\mathcal{I}, \mathcal{J} \subseteq \{1, 2, \ldots, n\}$. The superscripts "$T$" and "$H$" refer to the transpose and the Hermitian transpose. We write $i = \sqrt{-1}$ and use $\mathrm{Re}(x)$, $\mathrm{Im}(x)$, $\bar{x}$, $\angle x$, and $|x|$ to denote the real part, imaginary part, conjugate, angle, and absolute value. The identity matrix is I and the vector-of-ones is $\mathbf{1}$; their sizes are inferred from context. The trace, rank, and column vectorization are denoted $\mathrm{tr}(X)$, $\mathrm{rank}(X)$, and $\mathrm{vec}(X)$. $X \succeq 0$ 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 [15]. 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

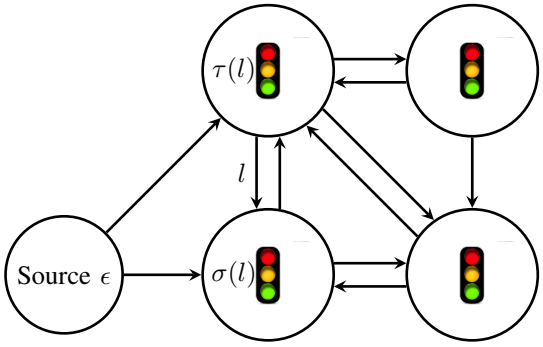

Fig. 1: Traffic Network

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 [23] 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 [15], 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 [23], 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

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

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

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

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 [15], 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:

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

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

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

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:

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

$$W_{j,k} = \sum_{l\in\mathcal{L}:\tau(l)=j,\sigma(l)=k} \bar{D}_l A_l + \sum_{l\in\mathcal{L}:\tau(l)=k,\sigma(l)=j} D_l \bar{A}_l \quad \text{for } j \neq k. \tag{7b}$$

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:

$$\begin{aligned} &\underset{z\in\mathbb{C}^n}{\text{maximize}} \; z^H W z \\ &\text{subject to } |z_j|^2 = 1, \quad j = 1, 2, \ldots, n. \end{aligned} \tag{8}$$

Given a solution $\hat{z}$ to the QCQP (8), one can obtain the optimal offsets of the traffic network via the equation

$$\theta_s = \frac{1}{2\pi}(\angle \hat{z}_s - \angle \hat{z}_n) \tag{9}$$

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

Remark 2. Note that the QCQP (8) formulated in this paper is subtly different from the one considered in [15]. Specifically, the diagonal elements of the matrix W in [15] are all zero so the matrix is not positive semidefinite. In our formulation, the matrix W in (8) 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 (8) 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 [24]. 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 [18]. Indeed, if the variable z in problem (8) is forced to be real, as in $z \in \mathbb{R}^n$, then the constraint $|z_j| = 1$ implies $z_j \in \{+1, -1\}$, and the maximization of a quadratic form subject to ± 1 variables is exactly MAX-CUT. Consequently, we may view (8) as a complex version of the MAX-CUT problem.

Based on the celebrated Goemans–Williamson algorithm [19] for MAX-CUT, we provide below a polynomial complexity algorithm that solves (8) 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 (8) 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:

$$\begin{aligned} &\underset{v_1,\ldots,v_n\in\mathbb{C}^n}{\text{maximize}} \; \sum_{j=1}^n \sum_{k=1}^n W_{j,k} v_j^H v_k \\ &\text{subject to } \|v_j\|^2 = 1, \quad j = 1, \ldots, n. \end{aligned} \tag{10}$$

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

$$\begin{aligned} &\underset{X\in\mathbb{C}^{n\times n}}{\text{maximize}} \; \mathrm{tr}(WX) \\ &\text{subject to } X_{j,j} = 1, \quad j = 1, \ldots, n, \\ &\qquad X \succeq 0. \end{aligned} \tag{11}$$

Lemma 3. *Problem* (11) *is a relaxation of* (8)*, and therefore, its value gives an upper-bound for the optimal value of* (8)*.*

Proof: Given any feasible solution $z \in \mathbb{C}^n$ of (8), 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 (11) and its objective value in (11) is the same as the objective value of z in (8). ∎

Problem (11) is an SDP for which an interior-point method is able to compute an optimal solution $\hat{X}$ 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 (10) 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 4. The SDP (11) can also be generated from (8) using a standard SDP relaxation procedure [25]. Specifically, by adding a rank constraint $\mathrm{rank}(X) = 1$ in (11), one obtain the original QCQP (8) because any rank-one matrix X can be factored into $X = zz^H$. The relaxation (11) becomes exact if its solution $\hat{X}$ has rank one. This special situation occurs for certain types of networks [26] and the offsets obtained from the SDP solution achieves optimal performance for these cases [15]. In general, however, the solution $\hat{X}$ of (11) 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 (10) back onto the one-dimensional unit sphere in $\mathbb{C}$ by randomized rounding

$$s_j = r^H \hat{v}_j, \qquad \hat{z}_j = s_j/|s_j|. \tag{12}$$

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

$$r = r_1 + ir_2, \qquad r_1, r_2 \sim \mathcal{N}(0, I) \tag{13}$$

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 5. *Given the optimal solution $\hat{v}_1, \ldots, \hat{v}_n \in \mathbb{C}^n$ for (10), define the candidate solution $\hat{z} \in C^n$ for* (8) *using (12) for each $\hat{z}_j \in \mathbb{C}$, in which $r \in \mathbb{C}^n$ is selected as in (13). Then,*

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

where $\textit{opt}_{QCQP}$ is the globally optimal value of (8) *and $\mathbb{E}[\cdot]$ is the expectation operator.*

Proof: The first bound is true because (10) is a relaxation of (8) by Lemma 3, and the second bound holds because $\hat{z}_1, \ldots, \hat{z}_n \in \mathbb{C}$ is not necessarily optimal for (8). The third bound follows from a result of [24], noting that $W \succeq 0$ from Lemma 1. ∎

In summary, this section describes a $\pi/4$-approximation algorithm for the QCQP (8) of the offset optimization problem that comprises two key steps:

1) Solve the SDP relaxation (11) 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 (12).

Standard algorithms implement these two 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* [22], 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 6. 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 7 ([22])**.** The *width* of a tree decomposition $(\mathcal{I}, T)$ is $\omega - 1$ where

$$\omega = \max_j \quad |\mathcal{I}_j|, \tag{14}$$

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 many problems related to a graph. The problem of computing the exact treewidth of a graph is known to be NP-complete [27]. 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 [28]. 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 [29]. 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

$$LL^H = PWP^H, \quad L \text{ is lower-triangular}, \quad L_{j,j} \geq 0. \tag{15}$$

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

$$\mathcal{I}_j = \{k \in \{1, \ldots, n\} : L_{k,j} \neq 0\}. \tag{16}$$

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

$$p(j) = \begin{cases} j & |\mathcal{I}_j| = 1, \\ \min_i\{i > j : i \in \mathcal{I}_j\} & |\mathcal{I}_j| > 1. \end{cases} \tag{17}$$

Lemma 8. *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 [30], the pair $(\mathcal{I}, T)$ forms a tree decomposition of W. From the definition (7) 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 [20] 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 (11) into a reduced-complexity problem with the variables $X_j \in \mathbb{C}^{|\mathcal{I}_j| \times |\mathcal{I}_j|}, j = 1, \ldots, n$:

$$\begin{aligned} \underset{X_1, \ldots, X_n}{\text{minimize}} \ & \sum_{j=1}^{n} \operatorname{tr}(W_j X_j) \\ \text{subject to } & (X_j)_{k,k} = 1, \quad j = 1, \ldots, n, \quad k = 1, \ldots, |\mathcal{I}_j| \\ & R_{p(j),j}(X_j) = R_{j,p(j)}(X_{p(j)}), \\ & X_j \succeq 0, \quad j = 1, \ldots, n, \end{aligned} \tag{18}$$

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 [21]. 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:

$$R_{k,j}(X_{\mathcal{I}_j, \mathcal{I}_j}) = X_{\mathcal{I}_k \cap \mathcal{I}_j, \mathcal{I}_k \cap \mathcal{I}_j} = R_{j,k}(X_{\mathcal{I}_k, \mathcal{I}_k}). \tag{19}$$

The associated constraints $R_{p(j),j}(X_j) = R_{j,p(j)}(X_{p(j)})$ in (18) 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 (11) to $O(\omega n)$ for $\{X_j, j = 1, \ldots, n\}$ in (18).

Lemma 9. *The solutions $\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n$ of* (18) *are related to the solution $\hat{X}$ of* (11) *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 [20]. ∎

The above relation allows us to recover the solution $\hat{X}$ of (11) from solutions $\hat{X}_1, \ldots, \hat{X}_j$ of (18). 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 (15). This can be done by the following Algorithm 1 adopted from [30].

Algorithm 1 Positive semidefinite matrix completion

Input. The column index sets $\mathcal{I}_1, \ldots, \mathcal{I}_n$ defined in (16) and the solutions $\hat{X}_1, \ldots, \hat{X}_j$ to (18).

Output. The solution $\hat{X}$ to (11) 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 (12). Specifically, from $\hat{X} = F^{-H}DF^{-1} = \hat{V}^H\hat{V}$ we obtain $\hat{V} = D^{1/2}F^{-1}$. Then, (12) is equivalent to

$$F^H s = D^{1/2} r, \qquad \hat{z}_j = s_j/|s_j| \tag{20}$$

where we recall that the real and imaginary parts of the random vector $r \in \mathbb{C}^n$ are selected independently and identically from the n-dimensional Gaussian distribution. Since F is a lower-triangular matrix (with the same sparsity pattern as L), one can compute $\hat{z}$ from (20) 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 (8) of the offset optimization problem given a tree decomposition with $\omega = O(1)$. The main steps are described as follows:

1) Reformulate (11) into the reduced complexity problem (18).
2) Solve (18) to obtain solutions $\hat{X}_1, \ldots, \hat{X}_n$.
3) Recover the solution of (11) 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 (20).

We will show later that the complexity of the overall algorithm is dominated by Step 2, i.e., the cost of solving the semidefinite program (18). An interior-point method solves (18) in $O(\sqrt{n})$ iterations, with the cost of each iteration dominated by the solution of a set of linear equations over $O(n)$ variables. These equations can be fully dense despite sparsity in the original problem, so the worst-case complexity of an interior-point solution of (18) is $O(n^{3.5})$ time and $O(n^2)$ memory. Next, we show that these complexity figures can be reduced to linear by using dualization to exploit sparsity.

C. Dualization

A recent result of [21] shows that the complexity of solving the real-valued version of (18) 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 [21] for the traffic offset optimization problem.

To solve (18), we begin by putting (18) into primal canonical form:

$$\begin{aligned}
\underset{x_1,\ldots,x_n \in \mathbb{C}^{n^2}}{\text{minimize}} \quad & \sum_{j=1}^{n} \bar{w}_j^H x_j \\
\text{subject to} \quad & \begin{bmatrix} N_{11} & \cdots & N_{1n} \\ & \ddots & \\ N_{n1} & \cdots & N_{nn} \\ M_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}, \\
& x_j \in \mathcal{K}_j, \quad j = 1, \ldots, n.
\end{aligned} \tag{21}$$

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 (19). 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 (21) 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 [21] for an example). Therefore, it takes $O(n^{3.5})$ time and $O(n^2)$ memory to solve (21) 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 10. *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 [21]. 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 [30]. The dualized problem of (21), posed in dual canonical form, is given by:

$$\begin{aligned}
\underset{y_1,\ldots,y_n \in \mathbb{C}^{n^2}}{\text{maximize}} \quad & -\sum_{j=1}^{n} \bar{w}_j^H y_j \\
\text{subject to} \quad & \begin{bmatrix} N_{11} & \cdots & N_{1n} \\ & \ddots & \\ N_{n1} & \cdots & N_{nn} \\ M_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} + s_0 = \begin{bmatrix} 0 \\ \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}, \\
& -y_j + s_j = 0, \quad j = 1, \ldots, n \\
& s_0 \in \{0\}^{n+1}, \quad s_j \in \mathcal{K}_j.
\end{aligned} \tag{22}$$

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

Since (22) 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 11. *A general-purpose interior-point method solves the SDP* (18) *by solving its dual canonical form* (22) *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 [21] 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 (22) 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 10, 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 (8) 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 (15), and determine the index sets $\mathcal{I}_1, \ldots, \mathcal{I}_n$ and the parent pointers p, as in (16) and (17).
3) Use the clique tree conversion technique to reformulate (11) into (18).
4) Convert (18) to the dualized problem (22).
5) Solve (22) as a dual canonical problem using a general-purpose interior-point method to obtain solutions $\hat{X}_1, \ldots, \hat{X}_n$ of (18).
6) Recover the solution of (11) in the sparse factored form $\hat{X} = F^{-H} D F^{-1}$ using Algorithm 1.
7) Recover a choice of $\hat{z}_1, \ldots, \hat{z}_n \in \mathbb{C}$ via the randomized rounding method (20). 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 12. *The proposed algorithm generates a choice of* $\hat{z}_1, \ldots, \hat{z}_n \in \mathbb{C}$ *that satisfy the bounds in Theorem 5 and can be computed with the same time and memory complexity as described in Theorem 11.*

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 11. 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 (20) 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 5. ∎

Remark 13. The offset optimization problem (8) 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, (8) is equivalent to

$$\begin{aligned} \underset{x,y \in \mathbb{R}^n}{\text{maximize}} \; & [x^H y^H] \begin{bmatrix} \text{Re}\,(W) & \text{Im}\,(W) \\ -\text{Im}\,(W) & \text{Re}\,(W) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ \text{subject to } & x_j^2 + y_j^2 = 1, \quad j = 1, 2, \ldots, n. \end{aligned} \tag{23}$$

One can then follow a similar procedure to solve this transformed real-valued problem as in our conference version [1]. However, transforming a complex QCQP into its real-valued counterpart also doubles its treewidth. In practice, the resulting algorithm is about a constant factor of 10 times slower than the one proposed in this paper. See [31] for such speed-up in optimization solvers using complex numbers instead of real numbers.

V. Numerical Experiments

In the previous sections, we proved that our algorithm solves offset optimization to a global optimality ratio of $\pi/4 \geq 0.785$ in near-linear $O(n^{1.5})$ time. In this section, we benchmark these guarantees on two datasets:

1) Realistic dataset for the Manhattan network, with real network topology, flow rates and turning ratios.
2) Synthetic dataset for the Berkeley, Manhattan, and Los Angeles networks, with real network topologies but synthetic flow rates and turning ratios.

In our numerical results described below, the empirical time complexity of the algorithm is linear $O(n)$, and the computed solutions have global optimality ratios exceeding 0.99.

A. Realistic Manhattan dataset

We demonstrate our algorithm in a real-world setting, by solving offset optimization on a realistic traffic model of Manhattan from Osorio et al. [32]. Our network graph contains 189 nodes and 472 edges, and covers the area between 7th and 12th Avenues, and 30th and 50th Streets. Detailed traffic simulations were performed to result in five sets of flow rates and turn ratios. In each case, green splits were assigned in order to make north-south links completely out of phase with east-west links.

We implement our algorithm in MATLAB and perform our experiments on a 3.3 GHz 4-core Intel Xeon E3-1230 v3 CPU with 16 GB of RAM. For each set of flow rates and turn ratios, we solve the convex relaxation to obtain a lower-bound ("lower"), and perform randomized rounding 200 times to obtain a suboptimal solution and an upper-bound ("upper"). As shown in Table I, all of the five optimization problems completed within ten seconds, to result in global optimality ratios of ≥ 0.996. We emphasize that global optimality

TABLE I: Offset optimization on the realistic Manhattan dataset

dataset	1	2	3	4	5	mean
upper	93591	104339	96160	107935	98639	100133
lower	93544	104267	96159	107740	98247	99991
ratio	0.9995	0.9993	1.0000	0.9982	0.9960	0.9986
sec	4.46	3.60	4.91	3.53	3.89	4.08

must be interpretted within the context of the formulation. In particular, they assume that traffic flow can be adequately approximated as being sinusoidal.

B. Synthetic OpenStreetMap dataset

To benchmark the scalability of our algorithm over a range of network sizes, we generate synthetic test cases using real-world network topologies collected from the OpenStreet Map data [33]. 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. 3a. The Berkeley-1 network has 405 intersections and 1122 links connecting the intersection, while the Berkeley-4 has 7000 intersections and 12176 links 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. 3b, and the third set of networks is based on the Downtown Los Angeles area as in Fig. 3c.

The network parameters and numerical results are presented in Table II. The number of intersections n ranges from 405 to 12176 among the networks in our experiments. In every case, the tree decomposition parameter ω is bounded by 50. The lower bound ("lower") on the squared queue length is the optimal value of the optimization problem (18) obtained from

TABLE II: Offset optimization on the synthetic Berkeley ("Berk"), Manhattan ("NYC"), and Los Angeles ("LA") datasets

Cases	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{L}\rvert$	ω	lower	upper	ratio	sec
Berk-1	405	1122	14	79209	79498	0.9964	6
Berk-2	2036	5789	36	477449	479725	0.9953	253
Berk-3	6979	19222	41	1588518	1597089	0.9946	1253
Berk-4	12176	33725	42	2795240	2810684	0.9945	2657
NYC-1	1430	2748	37	301366	303057	0.9944	234
NYC-2	2016	3854	31	417186	419692	0.9940	232
NYC-3	3923	7841	37	780878	787526	0.9916	655
NYC-4	9968	20945	39	2022529	2039907	0.9915	2565
LA-1	733	2180	22	182811	183403	0.9968	28
LA-2	1838	5170	36	458209	460708	0.9946	171
LA-3	3062	8838	43	747805	752536	0.9937	707
LA-4	4239	12773	50	1139072	1146237	0.9937	2207

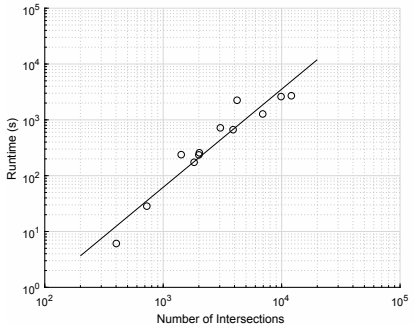

Fig. 2: Runtime against number of intersections. The regression line plots a *linear* empirical time complexity.

Step 5 of the algorithm. The optimal value of (18) serves as a bound according to Theorem 5. For each network, the upper bound ("upper") 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 MATLAB, 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 II, the performance of the algorithm is much better than the theoretical (worst-case) $\pi/4$ guarantee in Theorem 5. 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. 2 shows that the runtime scales almost linearly with respect to the number of intersections in the network. This agrees with the claim of Corollary 12 and it demonstrates the ability of our algorithm in solving large-scale traffic offset optimization.

VI. Conclusion

We describe an algorithm to solve a recently formulation [15] of the traffic signal offset optimization problem to near-global optimality in near-linear time. The algorithm

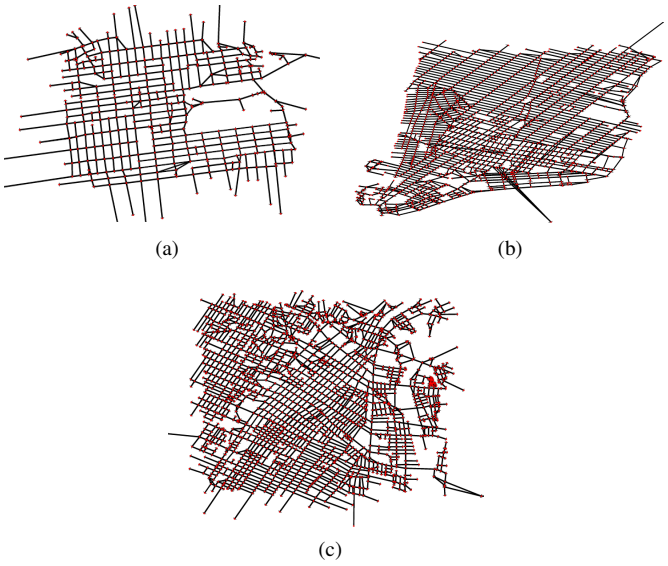

Fig. 3: The three road networks under study: (a) Berkeley; (b) Manhattan; (c) Downtown Los Angeles

performs a randomized rounding of an SDP relaxation to yield a suboptimal solution with global optimality bound $\pi/4 \geq 0.785$. Assuming that the traffic network has a "tree-like" topology, we prove that the algorithm has $O(n^{1.5})$ time complexity and $O(n)$ memory complexity with respect to the number of intersections n. 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 within an hour.

ACKNOWLEDGMENTS

The authors are grateful to Carolina Osorio and Timothy Tay at MIT for access to their realistic traffic models of Manhattan, and for their invaluable help in generating meaningful datasets.

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] N. H. Gartner, J. D. Little, and H. Gabbay, "Optimization of traffic signal settings by mixed-integer linear programming: Part ii: The network synchronization problem," *Transportation Science*, vol. 9, no. 4, pp. 344–363, 1975.

[3] N. Gartner, J. Little, and H. Gabbay, "Mitrop: a computer program for simultaneous optimisation of offsets, splits and cycle time," *Traffic Engineering & Control*, vol. 17, no. 8/9, 1976.

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

[5] N. H. Gartner, J. D. Little, and H. Gabbay, "Optimization of traffic signal settings by mixed-integer linear programming: Part I: The network coordination problem," *Transportation Science*, vol. 9, no. 4, pp. 321–343, 1975.

[6] C. E. Wallace, K. Courage, D. Reaves, G. Schoene, and G. Euler, *TRANSYT-7F user's manual*, 1984.

[7] Trafficware, *Synchro Studio 10 User Guide*, 2017.

[8] J. D. Little, "The synchronization of traffic signals by mixed-integer linear programming," *Operations Research*, vol. 14, no. 4, pp. 568–594, 1966.

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

[10] 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.

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

[12] 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.

[13] N. Gartner, "Optimal synchronization of traffic signal networks by dynamic programming," *Traffic Flow and Transportation*, 1972.

[14] S. Coogan, G. Gomes, E. S. Kim, M. Arcak, and P. Varaiya, "Offset optimization for a network of signalized intersections via semidefinite relaxation," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2187–2192.

[15] 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.

[16] Z. Amini, S. Coogan, C. Flores, A. Skabardonis, and P. Varaiya, "Optimizing offsets in signalized traffic networks: A case study," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2018, pp. 614–619.

[17] 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.

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

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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

[32] C. Osorio, X. Chen, J. Gao, M. Talas, and M. Marsico, "A scalable algorithm for the control of congested urban networks with intricate traffic patterns: New York City case studies," Tech. Rep., 2015.

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