Title	Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays
Author(s)	Leng, M; Wu, YC
Citation	IEEE Transactions on Signal Processing, 2011, v. 59 n. 10, p. 4860-4870
Issued Date	2011
URL	http://hdl.handle.net/10722/155657
Rights	©2011 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Low-Complexity Maximum-Likelihood Estimator for Clock Synchronization of Wireless Sensor Nodes Under Exponential Delays

Mei Leng and Yik-Chung Wu

***Abstract*—In this paper, the clock synchronization for wireless sensor networks in the presence of unknown exponential delay is investigated under the two-way message exchange mechanism. The maximum-likelihood estimator for joint estimation of clock offset, clock skew and fixed delay is first cast into a linear programming problem. Based on novel geometric analyses of the feasible domain, a low-complexity maximum likelihood estimator is then proposed. Complexities of the proposed estimators and existing algorithms are compared analytically and numerically. Simulation results further demonstrate that our proposed algorithms have advantages in terms of both performance and computational complexities.**

***Index Terms*—Clock synchronization, exponential delays, two-way message exchange mechanism, wireless sensor networks.**

I. Introduction

WIRELESS sensor networks have emerged recently as an important research area that structurally consists of many small-scale miniature devices (known as sensor nodes) capable of onboard sensing, computing and communications. WSNs are used in industrial and commercial applications to monitor data that would be difficult or inconvenient to monitor using wired equipment. These applications include monitoring the health status of environment, controlling industrial machines and home appliances, fire detection and object tracking, etc. [1], [2]. Most of these applications require collaborative execution of a distributed task amongst a large set of synchronized sensor nodes. Furthermore, data fusion, power management and transmission scheduling require all the nodes running on a common time frame. However, every individual sensor in a WSN has its own clock. Different clocks will drift from each other with time due to many factors, such as imperfection of the oscillators and environmental changes. This makes clock synchronization between different nodes an indispensable piece of infrastructure [3], [4].

Clock synchronization is not an easy task in practice due to several unique properties of WSN. The first and most important one is the limited power supply in low-end sensor nodes. Due to harsh operating conditions, nodes in WSNs are mostly left unattended for their lifetimes without any maintenance or battery replacement. To save power, each synchronization procedure should be simple and the frequency of re-synchronization should be low. This makes simplicity and accuracy the primary concerns of clock synchronization algorithms for WSNs.

The second challenge of clock synchronization in WSN is the unknown message delays in physical and MAC layers. Kopetz and Ochsenreiter [6] for the first time analyzed the process of message delay and decomposed the unknown delay into several components: send time, access time, transmission time, propagation time, reception time and receive time. These delay components can be grouped into two portions: the fixed delay and the random delay. The fixed delay is usually unknown, and if it is not modeled explicitly, it will be treated as a part of time offset, thus lowering the accuracy of timing parameter estimation. On the other hand, the random delay has been modeled as random variables following different distributions (such as Gaussian distribution, exponential distribution, Gamma and Weibull distribution) based on different justifications and applications. The difficulty of designing an optimal algorithm for clock synchronization largely depends on the modeling of this random delay.

When the random delay follows a Gaussian distribution, the optimal estimator for clock offset and clock skew in the presence of non-zero fixed delay has been given in [8] and [9]. However, as pointed out in [10], in many cases, e.g., when the point-to-point Hypothetical Reference Connection topology is of interest, the link delay between two nodes is appropriately represented as a regular M/M/1 queue, and the random delay should be modeled as exponential random variables. And in this case, it is much more difficult to design the optimal clock synchronizer.

Under exponential random delays, the joint estimation of clock offset and fixed delay is extensively studied in [11]–[14]. Unfortunately, the clock skew is not considered in these works, resulting in potentially frequent re-synchronization. On the other hand, the joint estimation of clock skew and fixed delay is studied in [15] with the clock offset left uncompensated, and the scheme is effective for clock synchronization when the peer clocks are initially started at the same time, i.e., the clock offset is zero. Apparently, by considering the fixed delay as a part of the clock offset, the algorithm in [15] can be adopted to jointly estimate clock skew and clock offset under the assumption that there is no fixed delay. In practice, however, the fixed delay is

Manuscript received September 17, 2010; revised March 20, 2011; accepted June 13, 2011. Date of publication June 27, 2011; date of current version September 14, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Sofiene Affes.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: meileng@eee.hku.hk; ycwu@eee.hku.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2011.2160857

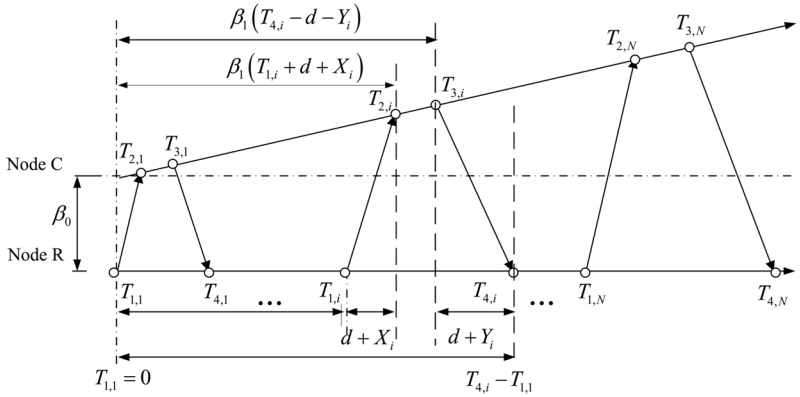


Fig. 1. Two-way time-stamps exchange between two nodes R and C.

usually non-neglegible. For example, as part of the fixed delay, the transmission/reception time can vary between 10 and 20 ms [16]. Note also that the required clock accuracy in WSNs is usually at microsecond (μs) order, the fixed delay can largely affect the estimation accuracy of the clock offset. Therefore, the algorithm in [8] relaxes the assumption of zero fixed delay and addresses the joint estimation of clock offset and clock skew by treating the fixed delay as a nuisance parameter. Unfortunately, since not all the available data are used, the algorithm is not optimal. Recently, a joint estimator maximizing the likelihood function for all three parameters is proposed in [17]. This estimator relies on an extension of a joint estimation scheme for clock skew and fixed delay, and assumes knowledge of clock offset. Although the estimator can achieve better performance, its complexity is high. Another recent work [19] solves the joint estimation problem of all three parameters by maximizing the profile likelihood function in terms of clock skew and finding the optimal solution through a one-dimensional search within a bounded interval. However, its performance and complexity depend on the user-defined searching resolution and interval.

In order to obtain the time information between two nodes, both one-way and two-way message exchange mechanisms have been proposed in the literature [3]–[5], [16]. However, in the case of jointly estimating clock skew, clock offset and fixed delay, it is shown in [18] that it is impossible to estimate the clock offset and fixed delay precisely with only one-way messages. And in order to eliminate the uncertainty caused by rank deficiency during estimation, a two-way message exchange mechanism is necessary.

In this paper, based on the two-way message exchange mechanism, a joint estimator of clock offset, clock skew and fixed delay is derived. The joint maximum likelihood estimation problem is first cast into a linear programming (LP) problem. Although the solution of the LP problem is guaranteed to be the globally optimal solution, directly solving this LP problem is not efficient. With novel geometric analyses of the feasible domain, an equivalent maximum-likelihood estimator (MLE) with much lower complexity is proposed. The rest of this paper is organized as follow. The system model is first introduced in Section II, and the MLE is cast as an LP-problem. In Sections III and IV, the feasible domain in which the optimal solution of the LP lies is analyzed. Based on the conducted analysis, an estimator with low complexity is proposed in Section V. Simulation results are presented in Section VI to illustrate the performance and complexity of the proposed estimator, and finally conclusions are drawn in Section VII.

II. System Model and Maximum-Likelihood Estimator

We consider the synchronization between a reference node R and its child node C based on a two-way timing message exchange mechanism as shown in Fig. 1. In the ith round of message exchange, node R sends a synchronization message to node C at $T_{1,i}$. Node C records its time $T_{2,i}$ at the reception of that message, and replies node R at $T_{3,i}$. The replied message contains both time-stamps $T_{2,i}$ and $T_{3,i}$. Then node R records the reception time of node C's reply as $T_{4,i}$. Note that $T_{1,i}$ and $T_{4,i}$ are the time stamps recorded by the clock of node R, while $T_{2,i}$ and $T_{3,i}$ are recorded by that of node C. After N rounds of message exchange, node C obtains a set of time stamps $\{T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i}\}_{i=1}^{N}$. The above procedure can be modeled as [17]

$$T_{2,i} = \beta_1 \cdot T_{1,i} + \beta_0 + \beta_1 \cdot (d + X_i) \tag{1}$$

$$T_{3,i} = \beta_1 \cdot T_{4,i} + \beta_0 - \beta_1 \cdot (d + Y_i) \tag{2}$$

where β_0 and β_1 represents the clock offset and clock skew of node C with respect to node R, respectively; d stands for the fixed portion of message delay from one node to another, and therefore is always non-negative; and X_i and Y_i are variable portions of the message delay. Based on the reasons explained in Section I, X_i and Y_i are modeled as independent and identical distributed (i.i.d.) exponential random variables with a common rate parameter λ, which is the most likely case in wireless applications, and is especially true when QoS delay control is implemented [10]. The goal is to estimate clock offset β_0 and clock skew β_1 based on the observation of a set of time-stamps $\{T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i}\}_{i=1}^{N}$.

To derive the MLE, we rewrite (1) and (2) as

$$X_i = \frac{1}{\beta_1} \cdot T_{2,i} - T_{1,i} - \frac{\beta_0}{\beta_1} - d \tag{3}$$

$$Y_i = -\frac{1}{\beta_1}\cdot T_{3,i} + T_{4,i} + \frac{\beta_0}{\beta_1} - d. \tag{4}$$

Since $\{X_i, Y_i\}_{i=1}^N$ are i.i.d. exponential random variables, denoting their probability density function as $\{p(X_i), p(Y_i)\}_{i=1}^N$, the likelihood function of $\{T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i}\}_{i=1}^N$ is given by $\prod_{i=1}^N p(X_i)p(Y_i)$. Substitute (3) and (4) into the likelihood function, we have the expression

$$\begin{aligned} &f\left(\{T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i}\}_{i=1}^N; \lambda, \theta_1, \theta_0, d\right) \\ &\quad = \lambda^{2N}\exp\left\{-\lambda\sum_{i=1}^N [(T_{2,i} - T_{3,i})\theta_1 - 2d + T_{4,i} - T_{1,i}]\right\} \\ &\qquad \times \prod_{i=1}^N I\left[T_{2,i}\theta_1 - \theta_0 - d - T_{1,i} \geq 0\right] \\ &\qquad \times \prod_{i=1}^N I\left[-T_{3,i}\theta_1 + \theta_0 - d + T_{4,i} \geq 0\right] \end{aligned} \tag{5}$$

where $\theta_0 \triangleq \frac{\beta_0}{\beta_1}$, $\theta_1 \triangleq \frac{1}{\beta_1}$, and $I[\cdot]$ is the indicator function. For given θ_1, θ_0 and d, the conditional MLE of λ can be obtained by differentiating the logarithm of (5) with respect to λ and setting the result to zero. It follows that

$$\hat{\lambda} = \frac{2N}{\sum_{i=1}^N [(T_{2,i} - T_{3,i})\theta_1 - 2d + T_{4,i} - T_{1,i}]}.$$

Putting $\hat{\lambda}$ back into (5) and discarding some irrelevant constants, we express the profile likelihood function [20] for $\{\theta_0, \theta_1, d\}$ as

$$\begin{aligned} &f(\{T_{1,i}, T_{2,i}, T_{3,i}, T_{4,i}\}_{i=1}^N; \theta_1, \theta_0, d) \\ &\quad \propto \left\{\sum_{i=1}^N [(T_{2,i} - T_{3,i})\theta_1 - 2d]\right\}^{-2N} \\ &\qquad \times \prod_{i=1}^N I\left[T_{2,i}\theta_1 - \theta_0 - d - T_{1,i} \geq 0\right] \\ &\qquad \times \prod_{i=1}^N I\left[-T_{3,i}\theta_1 + \theta_0 - d + T_{4,i} \geq 0\right]. \end{aligned} \tag{6}$$

Since MLE that maximizes the full likelihood function is equivalent to that maximizes the profile likelihood function [19], [20], we can find the optimal θ_0, θ_1 and d (denoted as $[\theta_0^*, \theta_1^*, d^*]$) by maximizing the profile likelihood function (6). Furthermore, notice that from the invariance property [21], the MLE of β_0, β_1 and d is equivalent to that of θ_0, θ_1 and d, since they are related by an invertible one-to-one transformation. Similar to the joint estimation of clock skew and fixed delay only using one-way message dissemination in [15], the MLE that maximizes (6) is equivalent to the solution of the following LP problem

$$\begin{aligned} [\theta_0^*, \theta_1^*, d^*] = \underset{\theta_1,\theta_0,d}{\operatorname{argmax}} &\sum_{i=1}^N [(T_{3,i} - T_{2,i})\theta_1 + 2d] \\ \text{subject to } &\begin{cases} \theta_0 - T_{3,1}\theta_1 + T_{4,1} - d \geq 0 \\ \vdots \\ \theta_0 - T_{3,N}\theta_1 + T_{4,N} - d \geq 0 \\ \theta_0 - T_{2,1}\theta_1 + T_{1,1} + d \leq 0 \\ \vdots \\ \theta_0 - T_{2,N}\theta_1 + T_{1,N} + d \leq 0 \\ d \geq 0. \end{cases} \end{aligned} \tag{7}$$

Since constraints in (7) define a feasible domain $\mathbb{S}$ which depends on unknown parameters θ_1, θ_0, and d, there is no simple closed-form solution. However, it can be solved using different numerical techniques, such as the simplex method or the interior-point method, and the solution is guaranteed to be globally optimum. Unfortunately, the numerical methods follow standard procedures to search for the optimal solution over the domain $\mathbb{S}$, and direct application is computationally expensive, especially for low-cost sensor nodes. If special structures of the constraints in (7) are taken into consideration, the construction of the domain $\mathbb{S}$ can be significantly simplified, which leads to an MLE with lower complexity.

From the geometric point of view, each constraint (i.e., inequality) in (7) spans a half-space in the three dimensional space of θ_1, θ_0 and d, and the domain $\mathbb{S}$ is a polyhedron defined by intersections of all the $2N+1$ half-spaces. Since the optimal solution of a LP problem must occur at a vertex [22], it is sufficient to consider only vertices of $\mathbb{S}$. For convenience, all the constraints in (7) are divided into two subsets with similar structures, and the LP problem is rewritten as

$$\begin{aligned} [\theta_0^*, \theta_1^*, d^*] = \underset{\theta_1,\theta_0,d}{\operatorname{argmax}} &\sum_{i=1}^N [(T_{3,i} - T_{2,i})\theta_1 + 2d] \\ \text{subject to} &\begin{cases} \mathcal{C}_1 \triangleq \{\{\theta_0 - T_{3,i}\theta_1 + T_{4,i} - d \geq 0\}_{i=1}^N; d \geq 0\} \\ \mathcal{C}_2 \triangleq \{\{\theta_0 - T_{2,i}\theta_1 + T_{1,i} + d \leq 0\}_{i=1}^N; d \geq 0\}. \end{cases} \end{aligned} \tag{8}$$

In the following, vertices defined by $\mathcal{C}_1$ and $\mathcal{C}_2$ when $d = 0$ are first considered, followed by analysis of vertices defined by $\mathcal{C}_1$ and $\mathcal{C}_2$ when $d > 0$.

III. Vertices of $\mathbb{S}$ When $d = 0$

With $d = 0$, $\mathcal{C}_1$ and $\mathcal{C}_2$ reduce to

$$\begin{aligned} \mathcal{C}_1' &\triangleq \{\theta_0 \geq T_{3,i}\theta_1 - T_{4,i}\}_{i=1}^N \\ \mathcal{C}_2' &\triangleq \{\theta_0 \leq T_{2,i}\theta_1 - T_{1,i}\}_{i=1}^N. \end{aligned}$$

Geometrically, these constraints define a feasible region in the θ_0-θ_1 plane which *lies above* all the supporting lines $\{\theta_0 = T_{3,i}\theta_1 - T_{4,i}\}_{i=1}^N$ from $\mathcal{C}_1'$ as well as *lies below* all the supporting lines $\{\theta_0 = T_{2,i}\theta_1 - T_{1,i}\}_{i=1}^N$ from $\mathcal{C}_2'$. For example, in Fig. 2,

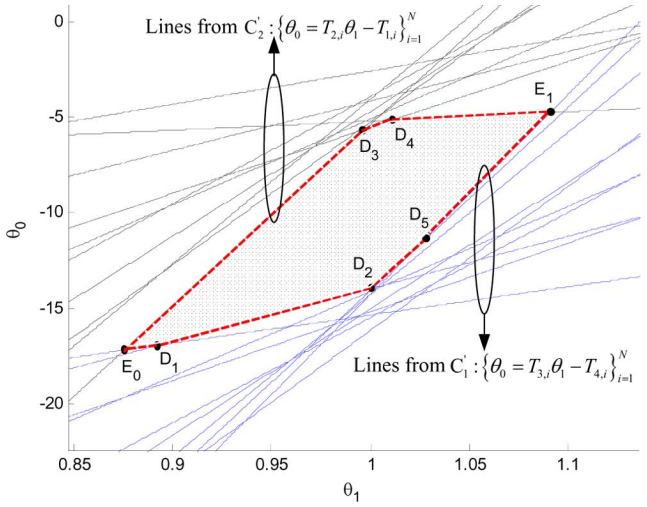


Fig. 2. Feasible region defined by $\mathcal{C}_1'$ and $\mathcal{C}_2'$ with $N = 10$.

the feasible region with $N = 10$ is the shaded polygon, and vertices are given by $\{E_0, D_1, D_2, D_3, D_4, D_5, E_1\}$. In order to find the vertices, boundaries of the feasible region should be determined first.

A. Boundaries From $\mathcal{C}_1'$ Only

To begin with, we consider boundaries from $\mathcal{C}_1'$ only. With $d = 0$, time-stamps $\{T_{3,i}, T_{4,i}\}_{i=1}^{N}$ can be interpreted as the timing information from one-way message dissemination from Node C to Node R, and the detection of boundaries from $\mathcal{C}_1'$ is similar to that in [15]. The sequential algorithm in [15] can be adopted here. For clarity, we first introduce the basic principle as follows.

After $N(N \geq 2)$ rounds of message exchanges, boundaries from $\mathcal{C}_1'$ are the *most-above* line-segments among $\{\theta_0 = T_{3,i}\theta_1 - T_{4,i}\}_{i=1}^{N}$. Suppose we have P such boundaries and denote them as $\{s_p : \theta_0 = T_{3,i_p}\theta_1 - T_{4,i_p}\}_{p=1}^{P}$, where $i_p \in \{1, \ldots, N\}$. When the $(N+1)$th round of message exchange finishes, we obtain one new supporting line in the θ_0-θ_1 plane:

$$u_{N+1} : \theta_0 = T_{3,N+1}\theta_1 - T_{4,N+1}.$$

The necessary condition for u_{N+1} to be a boundary is then

$$T_{3,N+1}\theta_1 - T_{4,N+1} \geq \{T_{3,i_p}\theta_1 - T_{4,i_p}\}_{p=1}^{P}$$

which is equivalent to

$$(T_{3,N+1} - T_{3,i_p})\theta_1 \geq T_{4,N+1} - T_{4,i_p}, \quad \forall\, p = 1, \ldots, P. \tag{9}$$

Defining

$$\mathcal{I} \triangleq \{i_p | T_{3,i_p} < T_{3,N+1}, p = 1, \ldots, P\}$$
$$\mathcal{I}^{\dagger} \triangleq \{i_p | T_{3,i_p} > T_{3,N+1}, p = 1, \ldots, P\}$$

we can rewrite the condition (9) as

$$\left\{\frac{T_{4,N+1} - T_{4,i_p}}{T_{3,N+1} - T_{3,i_p}}\right\}_{i_p \in \mathcal{I}} \leq \theta_1 \leq \left\{\frac{T_{4,N+1} - T_{4,i_p}}{T_{3,N+1} - T_{3,i_p}}\right\}_{i_p \in \mathcal{I}^{\dagger}}. \tag{10}$$

Defining

$$a_{N+1,i_p} \triangleq \frac{T_{4,N+1} - T_{4,i_p}}{T_{3,N+1} - T_{3,i_p}}$$

we can further simplify (10) as

$$\max_{i_p \in \mathcal{I}}\{a_{N+1,i_p}\} \leq \theta_1 \leq \min_{i_p \in \mathcal{I}^{\dagger}}\{a_{N+1,i_p}\}. \tag{11}$$

If (11) is satisfied for some θ_1, the new supporting line u_{N+1} is a boundary of the feasible region in the range indicated by (11). Otherwise, u_{N+1} has no effect and can be skipped.

The sequential algorithm in [15] provides an efficient method for checking whether a new line should be included in the boundary set of the feasible region. However, it assumes that $\{T_{3,i_p}\}_{p=1}^{P} < T_{3,N+1}$, implying $\mathcal{I}^{\dagger} = \emptyset$, and therefore only the left-hand side of (11) is checked. This is true in the one-way message dissemination, where $T_{3,i}$ is controlled by the transmitting node. However, in the two-way message exchange mechanism, $T_{3,i}$ is generated after receiving the i^{th} synchronization message from the reference node. Although $T_{1,i}$ is broadcasted periodically by the reference node, we cannot assume $T_{3,i-1} < T_{3,i}$ always hold since the ith synchronization message may be delayed due to random disturbance. In such cases, the sequential algorithm in [15] may fail.

Therefore, in the general case, we should first sort the time stamps $\{\{T_{3,i_p}\}_{p=1}^{P}, T_{3,N+1}\}$. Suppose

$$T_{3,i_1} < \cdots < T_{3,i_k} < T_{3,N+1} < T_{3,i_{k+1}} < \cdots < T_{3,i_P},$$

checking both sides of (11) is equivalent to finding the maximum among $\{a_{N+1,i_p}\}_{p=1}^{k}$ and the minimum among $\{a_{N+1,i_p}\}_{p=k+1}^{P}$. One straightforward way is to perform exhaustive comparison. On the other hand, we can narrow down the comparison using the following Lemma 1.

Lemma 1: For the left-hand side terms $\{a_{N+1,i_p}\}_{p=1}^{k}$, where $T_{3,i_p} < T_{3,N+1}$ holds, if $a_{N+1,i_p} > a_{i_{p-1},i_p}$, we have $a_{N+1,i_p} > a_{N+1,i_{p-1}} > \cdots > a_{N+1,i_1}$. For the right-hand side terms $\{a_{N+1,i_p}\}_{p=k+1}^{P}$, where $T_{3,N+1} < T_{3,i_p}$ holds, if $a_{N+1,i_p} < a_{i_p,i_{p+1}}$, we have $a_{N+1,i_p} < a_{N+1,i_{p+1}} < \cdots < a_{N+1,i_P}$.

Proof: See Appendix A. ∎

The implication of Lemma 1 is that, in order to find the maximum point on the left-hand side terms of (11), we should check $a_{N+1,i_p} > a_{i_{p-1},i_p}$ with p in decreasing order starting from $p = k$ until the inequality $a_{N+1,i_p} > a_{i_{p-1},i_p}$ holds. We denote the corresponding index as $\hat{p}$. Similarly, in order to find the minimum point on the right-hand side terms of (11), we should check $a_{N+1,i_p} < a_{i_p,i_{p+1}}$ with p in increasing order starting from $p = k+1$ until the inequality $a_{N+1,i_p} < a_{i_p,i_{p+1}}$ holds. We denote the corresponding index as $\hat{\hat{p}}$. Finally, if $a_{N+1,i_{\hat{p}}} < a_{N+1,i_{\hat{\hat{p}}}}$, the boundary set from $\mathcal{C}_1'$ can be updated

as $\{s_1,\ldots,s_{\hat{p}},u_{N+1},s_{\hat{p}},\ldots,s_P\}$. Otherwise, u_{N+1} has no effect on the boundary set and can be skipped. Based on the above discussions, we generalize the algorithm in [15] to sequentially detect boundaries from $\mathcal{C}_1'$, which is formally presented in the Algorithm 1.

Algorithm 1: Sequential Detection of Boundaries from $\mathcal{C}_1'$

1: Initialization at $N=2$: **if** $T_{3,1}<T_{3,2}$ **then** $i_1=1$ and $i_2=2$; **else** $i_1=2$ and $i_2=1$ **end if**

2: **for** each newly received supporting line $T_{3,N+1}\theta_1-T_{4,N+1}$ $u_{N+1}:\theta_0=$ **do**

3: Sort the time stamps such that $T_{3,i_1}<\cdots<T_{3,i_k}<T_{3,N+1}<T_{3,i_{k+1}}<\cdots<T_{3,i_P}$.

4: With $a_{i_{-1},i_0}\triangleq-\infty$, set $p=k$, check whether $a_{i_{p-1},i_p}<a_{N+1,i_p}$ holds. If it holds, set $\hat{p}=p$ and go to step 5. Otherwise, set $p=p-1$, and repeat step 4 until $p=0$.

5: With $a_{i_P,i_{P+1}}\triangleq+\infty$, set $p=k+1$, check whether $a_{i_p,i_{p+1}}>a_{N+1,i_p}$ holds. If it holds, set $\hat{\hat{p}}=p$ and go to step 6. Otherwise, set $p=p+1$, and repeat step 5 until $p=P+1$.

6: **if** $a_{N+1,i_{\hat{p}}}<a_{N+1,i_{\hat{\hat{p}}}}$ **then**

7: The boundaries set is updated as $\{s_1,\ldots,s_{\hat{p}},u_{N+1},s_{\hat{\hat{p}}},\ldots,s_P\}$.

8: **else**

9: The new supporting line $\theta_0=T_{3,N+1}\theta_1-T_{4,N+1}$ does not contribute to the boundaries from $\mathcal{C}_1'$ and can be skipped, therefore the boundaries set keeps unchanged as $\{s_p\}_{p=1}^P$.

10: **end if**

11: **end for**

B. Boundaries From $\mathcal{C}_2'$ Only

After N rounds of message exchanges, boundaries from $\mathcal{C}_2'$ are the *most-below* line-segments among $\{\theta_0=T_{2,i}\theta_1-T_{1,i}\}_{i=1}^N$. Suppose there are Q such boundaries denoted as $\{r_q:\theta_0=T_{2,j_q}\theta_1-T_{1,j_q},\}_{q=1}^Q$, where $j_q\in\{1,\ldots,N\}$, and $\{T_{2,N+1},T_{1,N+1}\}$ are obtained after the $(N+1)$th round of message exchange. Similar to (11), the new supporting line defined by $\{T_{2,N+1},T_{1,N+1}\}$ is a boundary of the feasible region if the following condition is satisfied,

$$\max_{j_q\in\{j_q|T_{2,j_q}>T_{2,N+1}\}}\underbrace{\left\{\frac{T_{1,N+1}-T_{1,j_q}}{T_{2,N+1}-T_{2,j_q}}\right\}}_{\triangleq b_{N+1,j_q}}\leq\theta_1$$
$$\leq\min_{j_q\in\{j_q|T_{2,j_q}<T_{2,N+1}\}}\left\{\frac{T_{1,N+1}-T_{1,j_q}}{T_{2,N+1}-T_{2,j_q}}\right\}.$$

Apparently, a procedure similar to Algorithm 1 can be employed to update the boundary set with knowledge of $\{\{T_{2,j_q},T_{1,j_q}\}_{q=1}^Q,T_{2,N+1},T_{1,N+1}\}$.

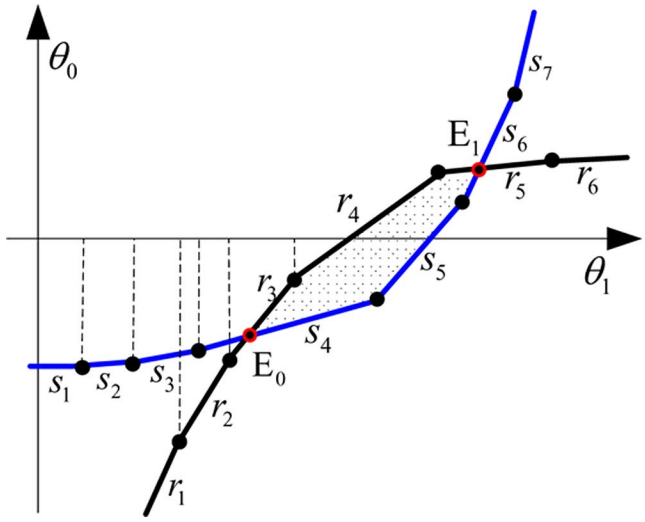


Fig. 3. Example of a feasible region defined by intersection of $\{s_p\}_{p=1}^P$ and $\{r_q\}_{q=1}^Q$.

C. Boundaries Defined by $\mathcal{C}_1'$ and $\mathcal{C}_2'$

Since one-way messages are not sufficient for precise estimation of clock offset and fixed delay, we must utilize all the information from the two-way message exchange, and obtain boundaries of the feasible region in the θ_0-θ_1 plane by combining $\{s_p\}_{p=1}^P$ and $\{r_q\}_{q=1}^Q$. For example in Fig. 3, the feasible region is the shaded polygon enclosed between $\{s_p\}_{p=1}^7$ and $\{r_q\}_{q=1}^6$. However, at the starting vertex E_0, the two starting boundaries are given by s_4 and r_3. Generally, denote the starting boundaries as s_{p_s} and r_{q_s}, we do not necessarily have $p_s=1$ and $q_s=1$. Similar observation can be obtained at the ending vertex E_1, and the ending boundaries are denoted as s_{p_e} and r_{q_e}. Therefore, boundaries of the feasible region are given by $\{s_p\}_{p=p_s}^{p_e}$ and $\{r_q\}_{q=q_s}^{q_e}$ with $1\leq p_s\leq p_e\leq P$ and $1\leq q_s\leq q_e\leq Q$.

In order to find $\{p_s,p_e,q_s,q_e\}$, it can be seen from Fig. 3 that E_0 and E_1 are the only two intersection points between $\{s_p\}_{p=1}^P$ and $\{r_q\}_{q=1}^Q$. In general, finding $\{p_s,q_s\}$ and $\{p_e,q_e\}$ is equivalent to finding indexes of the line-segments which form the first and second intersection point, respectively. Notice that two line-segments s_p and r_q can intersect only if they share a common range in the θ_1 direction. Therefore, we can divide the θ_1 axis into a number of intervals, according to θ_1-coordinates of the end-points on $\{s_p\}_{p=1}^P$ and $\{r_q\}_{q=1}^Q$, and check one interval after another.

More specifically, we first sort $\{a_{i_p,i_{p+1}}\}_{p=1}^{P-1}$ and $\{b_{j_q,j_{q+1}}\}_{q=1}^{Q-1}$ in ascending order, and denote the sorted sequence as $\{c_1,c_2,\ldots,c_{P+Q-2}\}$. In the interval $-\infty\leq\theta_1\leq c_1$, the corresponding line-segments must be s_1 and r_1. They intersect and form the starting boundaries if r_1 lies above s_1 at $\theta_1=c_1$. That is, if

$$T_{2,j_1}c_1-T_{1,j_1}\geq T_{3,i_1}c_1-T_{4,i_1}$$

holds. Otherwise, we check the interval $c_1\leq\theta_1\leq c_2$, where the two corresponding line-segments are $s_{p^{(1)}}$ and $r_{q^{(1)}}$ with the indexes given by

$$p^{(1)}=\underset{p\in\{1,\ldots,P-1\}}{\operatorname{argmin}}\{c_2\leq a_{i_p,i_{p+1}}\}$$
$$q^{(1)}=\underset{q\in\{1,\ldots,Q-1\}}{\operatorname{argmin}}\{c_2\leq b_{j_q,j_{q+1}}\}.$$

They intersect and form the starting boundaries if $r_{q^{(1)}}$ lies below $s_{p^{(1)}}$ at $\theta_1 = c_1$ and lies above $s_{p^{(1)}}$ at $\theta_1 = c_2$. That is, if

$$T_{2,j_{q^{(1)}}} c_1 - T_{1,j_{q^{(1)}}} < T_{3,i_{p^{(1)}}} c_1 - T_{4,i_{p_{(1)}}}$$
$$T_{2,j_{q^{(1)}}} c_2 - T_{1,j_{q^{(1)}}} > T_{3,i_{p^{(1)}}} c_2 - T_{4,i_{p_{(1)}}}$$

hold simultaneously. This procedure continues until we find the first intersection point, and it is formally given in Algorithm 2. The ending boundaries s_{p_e} and r_{q_e} can also be found using a similar procedure.

Algorithm 2: Finding $\{p_s, q_s\}$ With Knowledge of $\{s_p\}_{p=1}^{P}$ and $\{r_q\}_{q=1}^{Q}$

1: Combine $\{a_{i_p,i_{p+1}}\}_{p=1}^{P-1}$ and $\{b_{j_q,j_{q+1}}\}_{q=1}^{Q-1}$, and sort them in ascending order, denote the sorted sequence as $\{c_1, c_2, \ldots, c_{P+Q-2}\}$.
2: In the interval $-\infty \le \theta_1 \le c_1$, check whether

$$T_{2,j_1} c_1 - T_{1,j_1} \ge T_{3,i_1} c_1 - T_{4,i_1}$$

holds. If it holds, we have $p_s = 1$ and $q_s = 1$, and the procedure terminates. Otherwise, go to step 3.
3: Set $k = 1$.
4: In the interval $c_k \le \theta_1 \le c_{k+1}$, find $p^{(k)}$ and $q^{(k)}$ as

$$p^{(k)} = \underset{p\in\{1,\ldots,P-1\}}{\operatorname{argmin}} \{c_{k+1} \le a_{i_p,i_{p+1}}\}$$
$$q^{(k)} = \underset{q\in\{1,\ldots,Q-1\}}{\operatorname{argmin}} \{c_{k+1} \le b_{j_q,j_{q+1}}\}.$$

Check whether

$$T_{2,j_{q^{(k)}}} c_k - T_{1,j_{q^{(k)}}} < T_{3,i_{p^{(k)}}} c_k - T_{4,i_{p^{(k)}}}$$
$$T_{2,j_{q^{(k)}}} c_{k+1} - T_{1,j_{q^{(k)}}} > T_{3,i_{p^{(k)}}} c_{k+1} - T_{4,i_{p^{(k)}}}$$

hold simultaneously. If both inequalities hold, we have $p_s = p^{(k)}$ and $q_s = q^{(k)}$, and the procedure terminates. Otherwise, set $k = k + 1$, and repeat step 4.

D. Vertices Defined by $\mathcal{C}_1'$ and $\mathcal{C}_2'$

With knowledge of $\{s_p\}_{p=p_s}^{p_e}$ and $\{r_q\}_{q=q_s}^{q_e}$, vertices defined by $\mathcal{C}_1'$ and $\mathcal{C}_2'$ are intersection points of neighboring boundaries, and there are $p_e - p_s + q_e - q_s + 2$ vertices defined by $\mathcal{C}_1'$ and $\mathcal{C}_2'$. The first and last vertices, E_0 and E_1, are given by intersection of $\{s_{p_s}, r_{q_s}\}$ and $\{s_{p_e}, r_{q_e}\}$, respectively. Define the θ_1-coordinate of the intersection point between s_p and r_q as e_{i_p,j_q}, i.e.,

$$e_{i_p,j_q} \triangleq \frac{T_{4,i_p} - T_{1,j_q}}{T_{3,i_p} - T_{2,j_q}}.$$

We can express coordinates of E_0 and E_1 as

$$E_0 : \left[e_{i_{p_s},j_{q_s}},\ T_{3,i_{p_s}} e_{i_{p_s},j_{q_s}} - T_{4,i_{p_s}},\ 0\right] \quad (12)$$
$$E_1 : \left[e_{i_{p_e},j_{q_e}},\ T_{3,i_{p_e}} e_{i_{p_e},j_{q_e}} - T_{4,i_{p_e}},\ 0\right] \quad (13)$$

where entries of the tuple denote coordinates of θ_1, θ_0, and d, respectively.

Other vertices are given by intersections of neighboring boundaries among either $\{s_p\}_{p=p_s}^{p_e}$ or $\{r_q\}_{q=q_s}^{q_e}$. For vertices defined by intersection of s_p and s_{p+1}, denoted as $\{D_p^s\}_{p=p_s}^{p_e-1}$, the coordinates of D_p^s are

$$D_p^s : \left[a_{i_p,i_{p+1}},\ T_{3,i_p} a_{i_p,i_{p+1}} - T_{4,i_p},\ 0\right], \quad (14)$$

for $p = p_s, \ldots, p_e - 1$. On the other hand, for vertices defined by intersection of r_q and r_{q+1}, denoted as $\{D_q^r\}_{q=q_s}^{q_e-1}$, the coordinates of D_q^r are

$$D_q^r : \left[b_{j_q,j_{q+1}},\ T_{2,i_q} b_{j_q,j_{q+1}} - T_{4,i_q},\ 0\right], \quad (15)$$

for $q = q_s, \ldots, q_e - 1$.

Remark 1: In this section, the feasible domain with $d = 0$ is introduced. However, this cannot be generalized to the case when $d > 0$. Notice that with two parameters, the feasible domain is a 2-D polygon defined by lines. But with three parameters, the feasible domain is a 3-D polyhedron defined by planes, and the potential solution occurs at certain intersection point defined by three planes. With N rounds of message exchanges, there generally exist $\binom{2N+1}{3}$, i.e., $(4N^3 - N)/3$, intersection points. Algorithm 1 cannot be applied and direct searching from plane to plane is computationally prohibitive. For power-limited sensors, an algorithm with lower complexity is desirable. In the following, geometric analysis is conducted to show the relationship between the fixed delay and other two parameters. A low-complexity algorithm is then proposed based on the analysis.

IV. Vertices of $\mathbb{S}$ When $d > 0$

With $d > 0$, $\mathcal{C}_1$ and $\mathcal{C}_2$ become

$$\mathcal{C}_1'' \triangleq \left\{\{\theta_0 - T_{3,i}\theta_1 + T_{4,i} - d \ge 0\}_{i=1}^{N}; d > 0\right\},$$
$$\mathcal{C}_2'' \triangleq \left\{\{\theta_0 - T_{2,i}\theta_1 + T_{1,i} + d \le 0\}_{i=1}^{N}; d > 0\right\}.$$

To simplify $\mathcal{C}_1''$, we can make use of the results in the previous subsection when $d = 0$. In particular, for $a_{i_{p-1},i_p} \le \theta_1 \le a_{i_p,i_{p+1}}$, s_p lies above all other lines, that is,

$$T_{3,i_p}\theta_1 - T_{4,i_p} \ge \{T_{3,i}\theta_1 - T_{4,i}\}_{i=1,i\neq i_p}^{N}.$$

Adding $d - \theta_0$ on both sides, it follows that

$$\{\theta_0 - T_{3,i}\theta_1 + T_{4,i} - d\}_{i=1,i\neq i_p}^{N} \ge \theta_0 - T_{3,i_p}\theta_1 + T_{4,i_p} - d.$$

Therefore, the constraint set $\{\theta_0 - T_{3,i}\theta_1 + T_{4,i} - d \ge 0\}_{i=1}^{N}$ is dominated by $\theta_0 - T_{3,i_p}\theta_1 + T_{4,i_p} - d \ge 0$ for $a_{i_{p-1},i_p} \le \theta_1 \le a_{i_p,i_{p+1}}$. Since this is true for all p, $\mathcal{C}_1''$ is reduced to

$$\mathcal{C}_1'' = \{\{\theta_0 - T_{3,i_p}\theta_1 + T_{4,i_p} - d \ge 0\}_{p=p_s}^{p_e}; d > 0\}. \quad (16)$$

Similarly, $\mathcal{C}_2''$ can be reduced to

$$\mathcal{C}_2'' = \{\{\theta_0 - T_{2,j_q}\theta_1 + T_{1,j_q} + d \le 0\}_{q=q_s}^{q_e}; d > 0\}. \quad (17)$$

Notice from (16) and (17) that each inequality in $\mathcal{C}_1''$ and $\mathcal{C}_2''$ defines a half-space with a supporting plane[1], and each supporting plane has a base given by a line-segment from either

[1]Supporting plane: Given coefficients $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, for all $\mathbf{x}$ which satisfy $\mathbf{a}^T\mathbf{x} \ge b$ or $\mathbf{a}^T\mathbf{x} \le b$, the plane $\mathbf{a}^T\mathbf{x} = b$ is the supporting plane and serves as a boundary surface of the corresponding half-space [23].

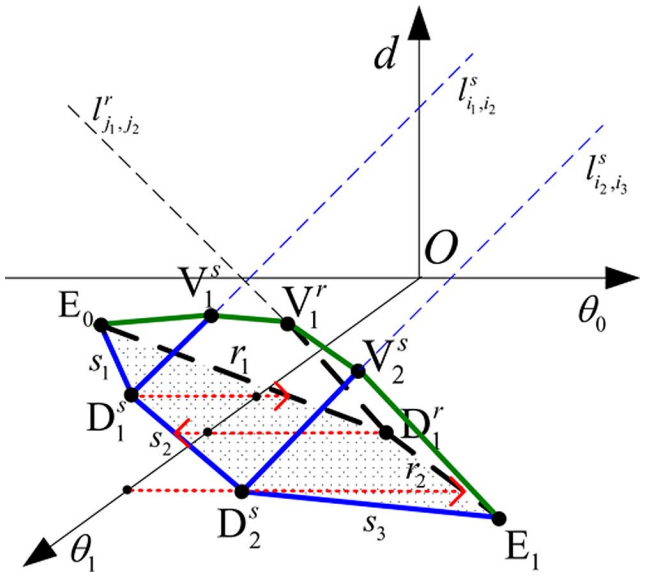


Fig. 4. Example of the domain $\mathbb{S}$; Vertices defined by $\mathcal{C}'_1$ and $\mathcal{C}'_2$: $\{E_0,\ D^s_1,\ D^s_2,\ D^r_1,\ E_1\}$ and Vertices defined by $\mathcal{C}''_1$ and $\mathcal{C}''_2$: $\{V^s_1, V^s_2, V^r_1\}$.

$\{s_p\}_{p=p_s}^{p_e}$ or $\{r_q\}_{q=q_s}^{q_e}$. Vertices defined by $\mathcal{C}''_1$ and $\mathcal{C}''_2$ are intersection points of any three such supporting planes. For example, a simple case with $\{s_p\}_{p=1}^{3}$ and $\{r_q\}_{q=1}^{2}$ is shown in Fig. 4.

To figure out how vertices defined by $\mathcal{C}''_1$ and $\mathcal{C}''_2$ are formed, we start with an analysis of the intersection between the supporting planes. Notice that each supporting plane from $\mathcal{C}''_1$ has a limited range (for example in Fig. 4, the supporting plane corresponding to s_1 is limited in the range $e_{i_1,j_1} \le \theta_1 \le a_{i_1,i_2}$), only two neighboring supporting planes can intersect, and we denote the intersection line between the i_p^{th} and i_{p+1}^{th} supporting planes as $\{l^s_{i_p,i_{p+1}}\}_{p=p_s}^{p_e-1}$. Similarly, denote the intersection line of two supporting planes from $\mathcal{C}''_2$ as $\{l^r_{j_q,j_{q+1}}\}_{q=q_s}^{q_e-1}$. It is shown in Appendix B that these intersection lines have the following property.

Lemma 2 (Parallel Intersection Lines of Supporting Planes): All the lines $\{l^s_{i_p,i_{p+1}}\}_{p=p_s}^{p_e-1}$ are parallel to the d-θ_0 plane with direction vectors $[0,1,1]$, where entries of the tuple denote coordinates of θ_1, θ_0 and d, respectively. On the other hand, all the lines $\{l^r_{j_q,j_{q+1}}\}_{q=q_s}^{q_e-1}$ are parallel to the d-θ_0 plane with direction vectors $[0,-1,1]$.

Proof: See Appendix B. ∎

As a result, three supporting planes from either $\mathcal{C}''_1$ or $\mathcal{C}''_2$ alone do not intersect. On the other hand, since direction vectors of supporting planes from $\mathcal{C}''_2$ are $\left\{\left[-T_{2,j_q},\ 1,-1\right]\right\}_{q=q_s}^{q_e-1}$, we can obtain the cross product between $\left[-T_{2,j_q},\ 1,-1\right]$ and the direction vector of $l^s_{i_p,i_{p+1}}$ as

$$[-T_{2,j_q},\ 1,-1]\times[0,\ 1,\ 1]=0$$

which means that vertices can be formed between $l^s_{i_p,i_{p+1}}$ from $\mathcal{C}''_1$ and any supporting plane from $\mathcal{C}''_2$. Similarly, vertices can be formed between $l^r_{j_q,j_{q+1}}$ from $\mathcal{C}''_2$ and any supporting plane from $\mathcal{C}''_1$.

Let us focus on the former case. Since $l^s_{i_p,i_{p+1}}$ parallels to the d-θ_0 plane and its θ_1-coordinate is given by $a_{i_p,i_{p+1}}$, the line $l^s_{i_p,i_{p+1}}$ must intersect with a supporting plane from $\mathcal{C}''_2$ whose range covers the point $\theta_1 = a_{i_p,i_{p+1}}$. Denoting the index of such a supporting plane as $j_{q'}$, we have

$$q' = \underset{q=q_s,\ldots,q_e-1}{\operatorname{argmin}} \{q | a_{i_p,i_{p+1}} \le b_{j_q,j_{q+1}}\}.$$

Since the vertex is formed by intersection of $l^s_{i_p,i_{p+1}}$ with the $j_{q'}$th supporting plane, the coordinates of the vertex can be obtained by solving the following linear equations:

$$\begin{aligned}\theta_0 - T_{3,i_p}\theta_1 + T_{4,i_p} - d = 0\\ \theta_0 - T_{3,i_{p+1}}\theta_1 + T_{4,i_{p+1}} - d = 0\\ \theta_0 - T_{2,j_{q'}}\theta_1 + T_{1,j_{q'}} + d = 0.\end{aligned}$$

Simple computations lead to the coordinates of the vertex as

$$V^s_p : \theta_1 = a_{i_p,i_{p+1}} \tag{18a}$$

$$\theta_0 = \frac{1}{2}\left[\left(T_{2,j_{q'}}+T_{3,i_p}\right)a_{i_p,i_{p+1}} - \left(T_{1,j_{q'}}+T_{4,i_p}\right)\right] \tag{18b}$$

$$d = \frac{1}{2}\left[\left(T_{2,j_{q'}}-T_{3,i_p}\right)a_{i_p,i_{p+1}} - d\left(T_{1,j_{q'}}-T_{4,i_p}\right)\right]. \tag{18c}$$

Similarly, $l^r_{j_q,j_{q+1}}$ must intersect with a supporting plane from $\mathcal{C}''_1$ whose range covers the point $\theta_1 = b_{j_q,j_{q+1}}$. Denote the index of this supporting plane as $i_{p'}$, we have

$$p' = \underset{p=p_s,\ldots,p_e-1}{\operatorname{argmin}} \{p | b_{j_q,j_{q+1}} \le a_{i_p,i_{p+1}}\},$$

and coordinates of the vertex are computed to be

$$V^r_q : \theta_1 = b_{j_q,j_{q+1}} \tag{19a}$$

$$\theta_0 = \frac{1}{2}\left[\left(T_{2,j_q}+T_{3,i_{p'}}\right)b_{j_q,j_{q+1}} - \left(T_{1,j_q}+T_{4,i_{p'}}\right)\right] \tag{19b}$$

$$d = \frac{1}{2}\left[\left(T_{2,j_q}-T_{3,i_{p'}}\right)b_{j_q,j_{q+1}} - \left(T_{1,j_q}-T_{4,i_{p'}}\right)\right]. \tag{19c}$$

Therefore, there are $p_e - p_s + q_e - q_s$ vertices defined by $\mathcal{C}''_1$ and $\mathcal{C}''_2$, and their coordinates are given in (18a)–(c) and (19a)–(c).

V. Low-Complexity Maximum-Likelihood Estimator

In Sections III and IV, we obtained coordinates of all the vertices on the domain $\mathbb{S}$. Combining vertices obtained from cases $d = 0$ and $d > 0$, the number of vertices is $2(p_e - p_s + q_e - q_s) + 2$. Since for the LP problem in (8), the optimal solution always appear at a vertex, a straightforward way to obtain the optimal solution is to compute the objective function in (8) at the identified $2(p_e - p_s + q_e - q_s) + 2$ vertices.

However, we can show that the number of vertices to be compared can be further reduced as follow. Denoting the value of the objective function at the vertex $V = [\theta_1, \theta_0, d]$ as $\mathcal{F}(V)$, we have

$$\mathcal{F}(V) \triangleq \sum_{i=1}^{N} (T_{3,i} - T_{2,i})\,\theta_1 + 2Nd. \tag{20}$$

For vertices D^s_p and V^s_p, we notice from (14) and (18) that they share the same value of θ_1-coordinate, which is $a_{i_p,i_{p+1}}$, however, the d-coordinate of D^s_p is zero and must be smaller than

the positive d-coordinate of V_p^s. Since $\mathcal{F}(V)$ depends only on θ_1 and d, it can be seen that

$$\mathcal{F}(V_p^s) > \mathcal{F}(D_p^s), \quad \text{for} \quad p = p_s, \ldots, p_e.$$

By the same argument, it can also be shown that

$$\mathcal{F}\left(V_q^r\right) > \mathcal{F}\left(D_q^r\right), \quad \text{for} \quad q = q_s, \ldots, q_e,$$
$$\mathcal{F}\left(V_p^s\right) > \mathcal{F}(E_0).$$

Therefore, in order to find the optimal solution, we need to check only the vertices $\left\{E_1, \{V_p^s\}_{p=p_s}^{p_e-1}, \{V_q^r\}_{q=q_s}^{q_e-1}\right\}$, and the number of vertices is reduced to $p_e - p_s + q_e - q_s + 1$, which is always smaller than $2N$.

Finally, the optimal solution of (8) is

$$[\theta_0^*, \theta_1^*, d^*] = \underset{\theta_1, d}{\operatorname{argmax}} \left\{\mathcal{F}(E_1), \left\{\mathcal{F}(V_p^s)\right\}_{p=p_s}^{p_e-1}, \left\{\mathcal{F}(V_q^r)\right\}_{q=q_s}^{q_e-1}\right\}. \tag{21}$$

The algorithm is formally given in Algorithm 3.

Algorithm 3: Low-Complexity MLE

1: Find the boundaries sets, $\{s_p\}_{q=1}^{Q}$ and $\{r_q\}_{p=1}^{P}$, using Algorithm 1.
2: Find $\{p_s, p_e, q_s, q_e\}$ using Algorithm 2.
3: Find coordinates of $\{V_p^s\}_{p=p_s}^{p_e-1}$ using **(18)**, and find coordinates of $\{V_q^r\}_{q=q_s}^{q_e-1}$ using **(19)**.
4: Find the optimal solution $[\theta_0^*, \theta_1^*, d^*]$ using **(21)**.
5: $\beta_1^* = \frac{1}{\theta_1^*}$ and $\beta_0^* = \frac{\theta_0^*}{\theta_1^*}$.

To analyze the complexity, an analysis to evaluate the number of floating point operations (additions/multiplications) is required for all the algorithms of interest. The computational complexity of the proposed algorithm is dictated by the number of computations involved in the Algorithms 1 and 2. In particular, by denoting as $\mathcal{O}(\cdot)$, Algorithm 1 involves $\mathcal{O}(N)$ division operations in the worst case, while Algorithm 2 involves $\mathcal{O}(N)$ multiplication operations in the worst case. Therefore, our algorithm has the complexity $\mathcal{O}(N)$ in the worst case. And by the worst case, we mean that every supporting line becomes one boundary of the feasible polygon. However, it should be noticed that such worst case rarely happens in practice. For example in Fig. 2, only 7 out of 20 supporting lines are boundaries, and the complexity will be significantly reduced. On the other hand, for the algorithm in [17], the computational cost in one cycle is $\mathcal{O}(N^2)$, and it will iterate for N^2 cycles in the worst case, making the worst case complexity $\mathcal{O}(N^4)$. For the LP problem, the simplex method and the interior-point method are the most widely-used [23]. In the simplex method, the algorithm is shown to take exponential number of steps with each step of complexity $\mathcal{O}(N^3)$ [24]. And in the interior-point method, the complexity is $\mathcal{O}(N^{3.5}L)$ with L denoting the total number of bits in a binary representation of the coefficients in the LP problem [25], [26]. Therefore, our proposed algorithm has much lower complexity than the algorithm in [17] and solving LP directly. On the other hand, the algorithm in [19] finds the MLE solution via 1-dimensional line search over a bounded interval. Hence, its performance and complexity depend on the user-defined step size (denoted as ϵ) and interval range (denoted as l). Specifically, its complexity is proportional to the total number of points (which is given by l/ϵ) and operations carried out at each point. Numerical results in Section VI confirm that our proposed MLE can achieve the same performance as the algorithm in [19] but with lower complexity.

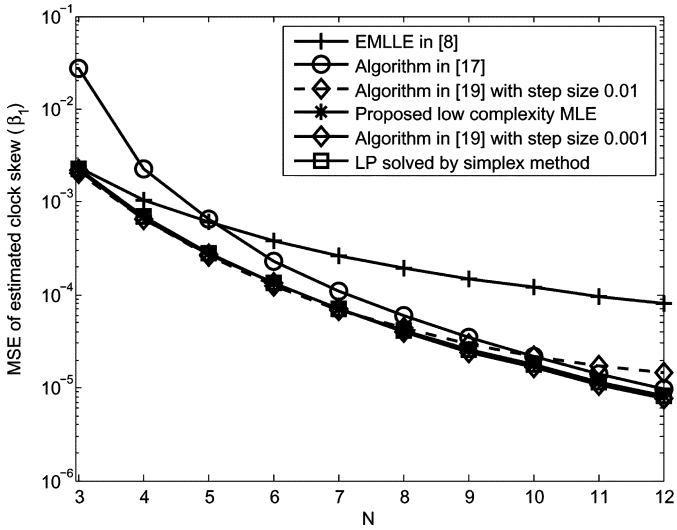


Fig. 5. MSE of estimated clock skew $\hat{\beta}_1$ with respect to the number of rounds of message exchange N.

VI. Simulation Results and Discussions

In this section, simulations are presented to validate the effectiveness of our proposed MLEs. The parameters to be estimated are uniformly drawn from ranges $\beta_1 \in (0.990, 1.010)$, $\beta_0 \in [-10, 10]$ and $d \in [1, 10]$. These parameter ranges are reasonable since the fixed delay and synchronization period are usually at order of millisecond (ms), and the deviation of clock skew, i.e., $|\beta_1 - 1|$, is generally on the order of 10^{-6} for a typical quartz clock [28]. Therefore, the order of $|\beta_1 - 1|$ should be 10^{-3} times those of β_0 and d. Other simulation settings are $\lambda = 1$, time difference between adjacent $T_{1,i}$ is 10, and the time waiting by Node C before replying is 5. Each point in the figures is an average of 10 000 independent simulation runs.

Fig. 5 shows MSE for estimation of the clock skew β_1 as a function of N. Besides the proposed algorithm, the LP solved by simplex method, the algorithm in [17], the algorithm in [19] with step sizes 0.001 and 0.01 (Since the objective function and constraints of the algorithm in [19] involve minimization functions, they are not twice continuously differentiable. Therefore, the problem is solved by direct search method), and the EMLLE algorithm in [8], are also simulated and compared. As shown in the figure, the algorithm in [19] with searching step size 0.001, the LP solver and the proposed low-complexity MLE all achieve the best performance. This is because they are all based on the same optimization problem. On the other hand, the algorithm in [19] with searching step size 0.01 exhibits degradation for large N, showing the sensitivity of this algorithm to the searching step size. Furthermore, the performance of the algorithm in [17] deteriorates slightly, because some rare special cases were not considered in [17]. However, its performance becomes very close to that of the optimal solution when N is large. For EMLLE,

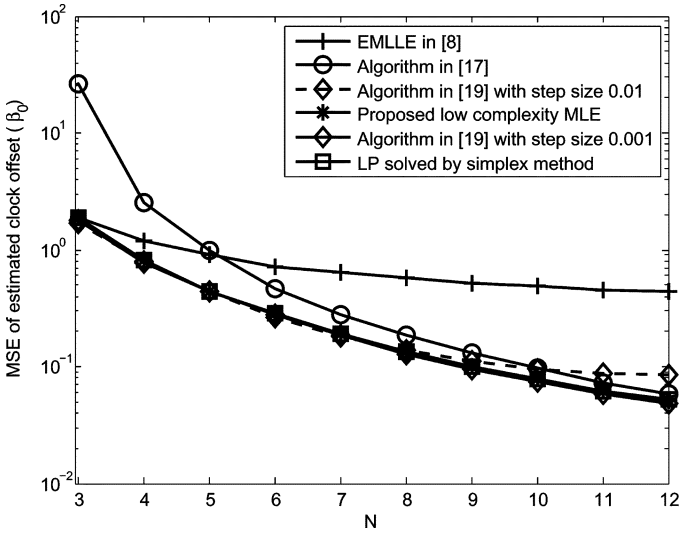


Fig. 6. MSE of estimated clock offset $\hat{\beta}_0$ with respect to the number of rounds of message exchange N.

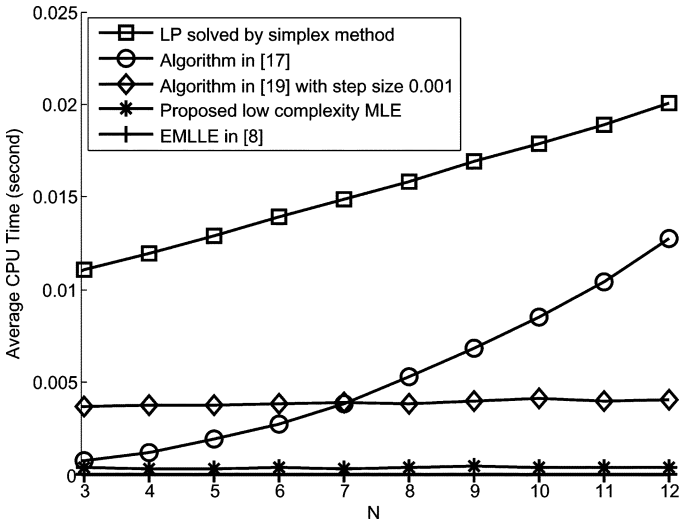


Fig. 7. Average CPU time cost with respect to the number of rounds of message exchange N.

since the estimation is not optimal, it presents a significant gap from other optimal algorithms. Fig. 6 shows the corresponding results for the clock offset β_0. It can be seen from the figure that the same conclusions as in Fig. 5 can be drawn.

In previous sections, we analytically analyzed the worst case complexities of different algorithms, showing the advantages of the proposed MLEs. However, in practice, the real-time complexities may be much smaller than the worst case complexities. Therefore, Fig. 7 shows the average CPU time of each estimator as a function of the number of round of message exchange N. Apparently, the estimator with less CPU time has lower complexity. As shown in the figure, the CPU time of the proposed low-complexity MLE is almost constantly at the order of 10^{-4} second and are significantly less than that of simplex method and the algorithm in [17] and [19]. On the other hand, the smallest complexities of EMLLE comes from the sacrifices of estimation performance as shown in previous two figures.

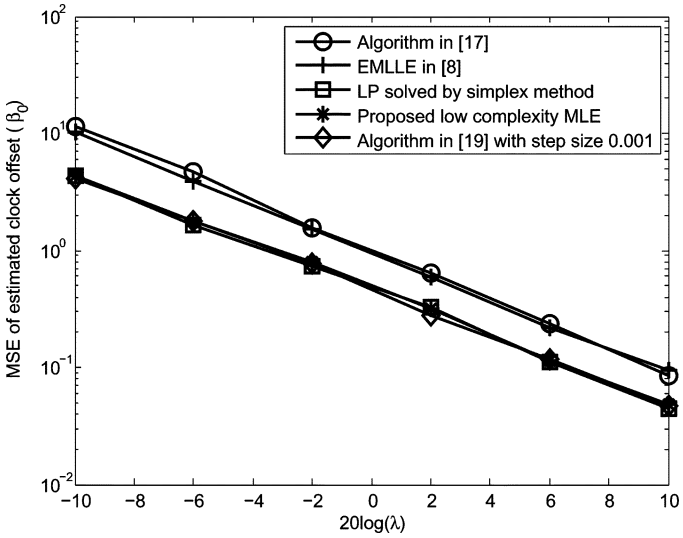


Fig. 8. MSE of estimated clock offset $\hat{\beta}_0$ with respect to the rate parameter λ.

Finally, MSE for estimation of the clock offset β_0 as a function of λ with $N = 5$ is shown in Fig. 8. Let the interval between adjacent $T_{1,i}$ as Δ, it can be easily shown that the signal-to-noise ratio (SNR) of estimation is proportional to $\Delta\lambda^2$. Therefore, increasing λ or increasing Δ has the same effect of increasing the SNR. From the figure, the LP solver, the proposed MLE and the algorithm in [19] with step size 0.001 provide the optimal solution and have the same performance, while the performance of the algorithm in [17] and EMLLE in [8] present a significant gap from that of optimal solution. Since the performance gap is constant for different λ, it can be concluded that the relative performance between different estimators is not affected by varying λ (or equivalently Δ). Furthermore, since the conclusions for the clock skew β_1 estimation are similar, we do not present the figure here.

VII. Conclusion

Clock synchronization for WSN in the presence of unknown exponential delay was investigated under the two-way message exchange mechanism. The MLE for joint estimation of clock skew, clock offset and fixed delay was first formulated as a linear programming problem. Although the solution provided by the linear programming solvers is guaranteed to be the global optimum, the computational complexity is high. Aiming at providing efficient solution, a low-complexity estimator was derived based on novel geometric analyses of the feasible domain defined by the constraints. The proposed MLEs obtain the solution at the vertex of the feasible domain with maximum objective value, and hence achieve the same performance as the linear programming solvers. Moreover, complexities of different estimators are compared analytically and numerically, and it is found that the proposed MLE is much simpler than simplex method in linear programming, the algorithms in [17] and [19]. Therefore, the proposed estimator represents an attractive time synchronization algorithm in terms of both computational complexity and performance.

APPENDIX A

Since terms on the left-hand and right-hand sides are similar, we prove only for the left-hand side terms here. Firstly, we show that if $T_{3,N+1} > T_{3,i_k}$ and $a_{N+1,i_k} > a_{i_{k-1},i_k}$, we have $a_{N+1,i_k} > a_{N+1,i_{k-1}}$. Since

$$\frac{T_{4,N+1}-T_{4,i_k}}{T_{3,N+1}-T_{3,i_k}} > \frac{T_{4,i_{k-1}}-T_{4,i_k}}{T_{3,i_{k-1}}-T_{3,i_k}}, \tag{22}$$

utilizing the fact that $T_{3,N+1} > T_{3,i_k} > T_{3,i_{k-1}}$, the following inequality follows directly from (22):

$$\begin{aligned} &T_{3,N+1}T_{4,i_{k-1}} - T_{3,N+1}T_{4,i_k} - T_{3,i_k}T_{4,i_{k-1}} \\ &\quad > T_{4,N+1}T_{3,i_{k-1}} - T_{4,N+1}T_{3,i_k} - T_{4,i_k}T_{3,i_{k-1}}. \end{aligned} \tag{23}$$

Multiplying by $T_{3,N+1}$ and adding the term $-T_{4,N+1}T_{3,i_{k-1}}T_{3,i_k}$ to both sides of (23), it follows that

$$\frac{T_{4,N+1}T_{3,i_k} - T_{3,N+1}T_{4,i_k}}{T_{3,N+1}-T_{3,i_k}} > \frac{T_{4,N+1}T_{3,i_{k-1}} - T_{3,N+1}T_{4,i_{k-1}}}{T_{3,N+1}-T_{3,i_{k-1}}}.$$

Notice that the above inequality is equivalent to

$$\begin{aligned} T_{3,N+1}\frac{T_{4,N+1}-T_{4,i_k}}{T_{3,N+1}-T_{3,i_k}} - T_{4,N+1} \\ > T_{3,N+1}\frac{T_{4,N+1}-T_{4,i_{k-1}}}{T_{3,N+1}-T_{3,i_{k-1}}} - T_{4,N+1} \end{aligned}$$

and based on the fact that $T_{3,i_{N+1}} > 0$, we have

$$\frac{T_{4,N+1}-T_{4,i_k}}{T_{3,N+1}-T_{3,i_k}} > \frac{T_{4,N+1}-T_{4,i_{k-1}}}{T_{3,N+1}-T_{3,i_{k-1}}},$$

which means $a_{N+1,i_k} > a_{N+1,i_{k-1}}$. Notice that $a_{i_{k-1},i_k} > \cdots > a_{i_1,i_2} > a_{i_0,i_1}$, other cases $a_{N+1,i_{k-1}} > a_{N+1,i_{k-2}} > \cdots > a_{N+1,i_1}$ can be proved in a similar way.

APPENDIX B
PROOF OF LEMMA 2

For the following two supporting planes from $\mathcal{C}_1''$:

$$\begin{aligned} \theta_0 - T_{3,i}\theta_1 + T_{4,i} - d = 0 \\ \theta_0 - T_{3,j}\theta_1 + T_{4,j} - d = 0 \end{aligned}$$

we have normal vectors $\mathbf{n}_i \triangleq [-1, 1, -T_{3,i}]$ and $\mathbf{n}_j \triangleq [-1, 1, -T_{3,j}]$, respectively. Notice that the cross product of $\mathbf{n}_i$ and $\mathbf{n}_j$ is given by

$$\begin{aligned} \mathbf{n}_i \times \mathbf{n}_j &= \det \begin{vmatrix} \theta_1 & \theta_0 & d \\ -T_{3,i} & 1 & -1 \\ -T_{3,j} & 1 & -1 \end{vmatrix} \\ &= [0, T_{3,j} - T_{3,i}, T_{3,j} - T_{3,i}] \\ &= [0, 1, 1], \end{aligned}$$

together with the fact that two planes intersect in a line if and only if $\mathbf{n}_i \times \mathbf{n}_j \neq \mathbf{0}$, it is confirmed that these two supporting planes always intersect in a line, and this line is denoted as $l^s_{i,j}$. Moreover, since $l^s_{i,j}$ lies in both planes, it must be perpendicular to both $\mathbf{n}_i$ and $\mathbf{n}_j$. Therefore, the direction vector of $l^s_{i,j}$ is given by the cross product $\mathbf{n}_i \times \mathbf{n}_j$ [27] and equals $[0, 1, 1]$, where entries of the tuple denote coordinates of θ_1, θ_0 and d, respectively. Notice that the θ_1-coordinate of the direction vector is zero, so $l^s_{i,j}$ is parallel to the θ_0-d plane.

Similarly, the same conclusion can be drawn for $\{l^r_{i,j}\}_{q=q_s}^{q_e-1}$. In particular, denote the intersection line between $T_{2,i}\theta_1 - \theta_0 - d - T_{1,i} = 0$ and $T_{2,j}\theta_1 - \theta_0 - d - T_{1,j} = 0$ as $l^r_{i,j}$, its direction vector is given by the cross product of the normal vectors $[T_{2,i}, -1, -1]$ and $[T_{2,j}, -1, -1]$, which can be easily computed to be $[0, -1, 1]$, and $l^r_{i,j}$ is also parallel to the θ_0-d plane.

Since i, j can take any value from $\{1, \ldots, N\}$, $\{l^s_{i_p,j_{p+1}}\}_{p=p_s}^{p_e-1}$ and $\{l^r_{j_q,j_{q+1}}\}_{q=q_s}^{q_e-1}$ also possess the above properties.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[2] N. Bulusu and S. Jha, *Wireless Sensor Networks: A Systems Perspective*. Norwood, MA: Artech House, 2005.

[3] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 3, pp. 281–323, 2005.

[4] S. Ganeriwal, R. Kumar, and M. B. Srivastava, *Timing-Sync Protocol for Sensor Networks*. Univ. of California, Los Angeles: Center for Embedded Network Sensing, 2003.

[5] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. 5th Symp. Operat. Syst. Des. Implement. Symp.*, Boston, MA, Dec. 2002, pp. 147–163.

[6] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *IEEE Trans. Comput.*, vol. 36, no. 8, pp. 933–940, Aug. 1987.

[7] B. M. Sadler, "Local and broadcast clock synchronization in a sensor node," *IEEE Signal Process. Lett.*, vol. 13, no. 1, Jan. 2006.

[8] K. L. Noh, Q. M. Chaudhari, E. Serpedin, and B. W. Suter, "Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 766–777, Apr. 2007.

[9] M. Leng and Y.-C. Wu, "On clock synchronization algorithms for wireless sensor networks with unknown delay," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 182–190, Jan. 2010.

[10] H. S. Abedel-Ghaffar, "Analysis of synchronization algorithms with time-out control over networks with experimentally symmetric delays," *IEEE Trans. Commun.*, vol. 50, no. 10, pp. 1652–1661, Oct. 2002.

[11] D. R. Jeske and A. Sampath, "Estimation of clock offset using bootstrap bias-correction techniques," *Technometrics*, vol. 45, no. 3, pp. 256–261, Aug. 2003.

[12] D. R. Jeske, "On the maximum likelihood estimation of clock offset," *IEEE Trans. Commun.*, vol. 53, pp. 53–54, Jan. 2005.

[13] Q. M. Chaudhari, E. Serpedin, and J. S. Kim, "Energy-efficient estimation of clock offset for inactive nodes in wireless sensor networks," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 582–596, Jan. 2010.

[14] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, "Some improved and generalized estimation schemes for clock synchronization of listening nodes in wireless sensor networks," *IEEE Trans. Commun.*, vol. 58, no. 1, pp. 63–67, Jan. 2010.

[15] T. Trump, "Maximum likelihood trend estimation in exponential noise," *IEEE Trans. Signal Process.*, vol. 49, no. 9, pp. 2087–2095, Sep. 2001.

[16] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, New York, 2004, pp. 39–49.

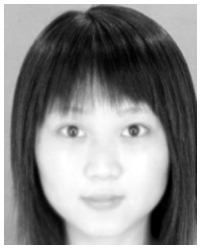
[17] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, "On maximum likelihood estimation of clock offset and skew in networks with exponential delays," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1685–1697, Apr. 2008.

[18] N. M. Freris and P. R. Kumar, "Fundamental limits on synchronization of affine clocks in networks," in *Proc. 46th IEEE Conf. Decision Control*, New Orleans, LA, Dec. 12–14, 2007, pp. 921–926.

[19] J. Li and D. R. Jeske, "Maximum likelihood estimators of clock offset and skew under exponential delays," *Appl. Stoch. Models Bus. Ind.*, vol. 25, pp. 445–459, 2009.

[20] S. A. Murphy and V. D. Vaart, "On profile likelihood," *J. Amer. Statist. Assoc.*, no. 95, pp. 449–465, 2000.

[21] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[22] S. Boyd and L. Vandenberghe, *Convex Optimizaton*, 7th ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[23] G. B. Dantzig and M. N. Thapa, *Linear Programming II—Theory and Extensions*. New York: Springer Series in Operations Research, 2003.
[24] D. A. Sielman and S. H. Teng, "Smoothed analysis—Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, no. 3, pp. 385–463, 2004.
[25] S. A. Vavasis and Y. Y. Ye, "A primal-dual interior point method whose running time depends only on the constraint matrix," *Math. Programm., Series A and B*, vol. 74, no. 1, pp. 79–120, Jul. 1996.
[26] N. Megiddo, "On the complexity of linear programming," in *Advances in Economic Theory: Fifth World Congress*, T. Bewley, Ed. Cambridge, U.K.: Cambridge Univ. Press, 1987, pp. 225–268.
[27] G. Strang, *Linear Algebra and its Applications*, 4th ed. Ocean, NJ: Nelson Engineering, 2007.
[28] F. Cristian, "Probabilistic clock synchronization," *Distrib. Comput.*, no. 3, pp. 146–158, 1989.



Mei Leng received the B.Eng. degree from the University of Electronic Science and Technology of China (UESTC) in 2005 and the Ph.D. degree from The University of Hong Kong in 2011.

She is currently a Research Fellow at the School of Electrical and Electronic Engineering, Nanyang Technological University. Her current research interests include statistical signal processing, optimization, machine learning as well as Bayesian analysis, with applications to wireless sensor networks and wireless communication systems.

Yik-Chung Wu received the B.Eng. (E.E.E.) degree in 1998 and the M.Phil. degree in 2001, both from The University of Hong Kong (HKU), and the Ph.D. degree from Texas A&M University, College Station, in 2005.

From August 2005 to August 2006, he was a Member of Technical Staff with the Thomson Corporate Research, Princeton, NJ. Since September 2006, he has been an Assistant Professor with the University of Hong Kong. His research interests are in general area of signal processing and communication systems, and in particular receiver algorithm design, synchronization techniques, channel estimation, and equalization.

Dr. Wu is currently serving as an Associate Editor for the IEEE COMMUNICATIONS LETTERS.