On the Hardness of Eliminating Cheating Behavior in Time Synchronization Protocols for Sensor Networks

Murtuza Jadliwala, Qi Duan, Shambhu Upadhyaya and Jinhui Xu
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260.
Email: {msj3, qiduan, shambhu, jinhui}@cse.buffalo.edu

Abstract

Wireless Sensor Networks are fast gaining popularity for use in a variety of remote sensing, emergency monitoring and information collection applications. Time synchronization in such highly distributed systems is important in order to maintain a global notion of time throughout the network and to support the underlying applications. But, cheating behavior by malicious nodes can severely jeopardize the accuracy of the time synchronization service associated with the network and the related time-based applications. Existing literature on the problem of detection and elimination of cheating behavior in distributed time synchronization protocols has very little or no fundamental theoretical analysis of the basic problem itself. Absence of sound theoretical models and analysis of time synchronization protocols in such networks has left a number of questions unanswered: How hard is it to detect and eliminate cheating or inconsistent behavior in time synchronization protocols given complete (local time and time difference) information? Do efficient algorithms for doing the same exist and if they do, is there a performance guarantee on the solution quality? In this paper, we attempt to answer these questions. We first present a practical graph-theoretic model, called Time Difference Graphs (TDG), for the network and formulate the time synchronization problem as a Constraint Satisfaction Problem (CSP) in such graphs. We then show that efficiently eliminating cheating behavior (or inconsistency causing nodes) in TDGs is *NP*-hard. Furthermore, we show that this problem is hard even for a special case of TDG, namely, a completely connected TDG. Moreover, we show that it is impossible to approximate this problem within a factor $n^{1-\varepsilon}$, where $\varepsilon > 0$, unless $P = NP$. Finally, we propose two polynomial time approximation strategies, namely a greedy strategy and an Integer Programming formulation, for efficiently eliminating inconsistency causing nodes in complete TDGs.

Keywords: Combinatorial Optimization, Graph Theory, Time Synchronization and Wireless Sensor Networks

1 Introduction

Recent advances in sensor, processor and radio technology has resulted in the integration of sensing, computation and wireless communication capabilities into a single, low form-factor and low cost system called a sensor mote. On deployment at the area of interest, these motes form a wireless ad hoc network without any infrastructure support and is referred to as a Wireless Sensor Network (WSN). WSNs are gaining popularity in outdoor monitoring and emergency response applications that include but not limited to environmental and climate monitoring, forest fire disaster recovery, target tracking and monitoring enemy movement during wars and intrusion detection systems. Due to the critical nature of these applications, the order and time of occurrence of the monitored events in such applications is extremely important. In order to achieve this, each mote maintains an on-chip local clock which at the time of deployment is synchronized with the other motes in the network. This local clock is nothing but a quartz crystal oscillating at a specific frequency and the notion of time is maintained by counting the number of oscillations of this quartz crystal. Over a period of time, motes in the network develop varying notions of time due to various network and mote dependent factors like fading battery power, longer sleep periods, crystals oscillating at different frequencies etc.

The process of updating the local clocks of every mote in the network such that all motes have the same notion of time is called *Time Synchronization*. Time synchronization can be an *Absolute Synchronization* or *Relative Synchronization*. In absolute synchronization, the clocks of all the motes are adjusted to a real-time standard like UTC (Universal Coordinated Time) or some other global value. In relative synchronization, such a global or standard value of time is not known and the nodes have to be synchronized relative to each other. Several efficient protocols for time synchronization in large computer networks (both internet and Internet) and other highly distributed systems exist in the literature [30]. Cristian's Remote Clock Reading method [6], Arvind's Time Transmission Protocol (TTP) [2], Lemmon et al.'s Set-valued Estimation method [17] and Mill's Offset Delay Estimation method employed by the Network Time Protocol (NTP) [20] are a few example of such protocols. These protocols estimate the time offset between the target machine (that needs to synchronize its time) and a source machine (generally a time-keeping server or a machine that has the correct notion of time) using simple message transmissions and time stamping, and use this estimated time difference to compute the time for the target. But due to lack of infrastructure, limited power, limited bandwidth, limited hardware and non-deterministic delays at the MAC layer in sensor motes the above schemes cannot be directly applied to sensor networks.

Depending on the application, either relative or absolute synchronization is required in sensor networks. But, for most applications a relative synchronization is sufficient. Relative time synchronization algorithms for wireless sensor networks are further divided into two broad types: *Sender-Receiver* synchronization and *Receiver-Receiver* synchronization [30]. In Sender-Receiver synchronization [12, 23, 26, 1] the nodes synchronize themselves with respect to a sender or beacon node. The sender node periodically sends a message (beacon) with its local time as a time-stamp to the receiver. The receiver then synchronizes with the sender using the time-stamp it receives from the sender. The message delay between the sender and receiver is calculated by measuring the total round-trip time, from the time a receiver requests a time-stamp until the time it actually receives a response. Receiver-Receiver synchronization [7, 28, 21, 19, 31, 3] exploits the property of the physical broadcast medium that if any two receivers receive the same message in single-hop transmission, they receive it at approximately the same time. Instead of interacting with a sender, receivers exchange the time at which they received the same message and compute their offset based on the difference in reception times. One thing that is common to all the time synchronization techniques mentioned above is that each node[†] that wants to synchronize itself attempts to estimate the time offset (difference) to some (or all) its neighboring nodes. The number and type of nodes that the target node estimates time differences to and the technique used to estimate this time difference is different for each scheme and depends on the time synchronization protocol used.

In the case that all the nodes honestly execute their protocols, the estimated time differences in the network should follow the triangle law as shown in Fig. 1. Figure 1 shows that if nodes A, B and C honestly

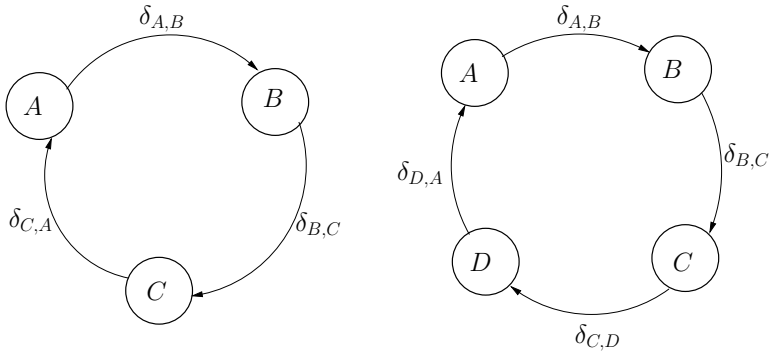

Figure 1: Triangle Law for Time Offset

compute their time offsets and if $\delta_{A,B}$ is the time offset of node A with respect to node B and $\delta_{B,C}$ is the time

[†]The term "mote" and "node" are used interchangeably

offset of node B with respect to node C and so on, then the sum of the time differences $\delta_{A,B}$, $\delta_{B,C}$ and $\delta_{C,A}$ is zero. In practice, this sum should be less than some constant ε, which denotes an application specific upper bound on the measurement error in the network. But without loss of generality, we can assume here that if nodes are honest then the sum of the above time differences is zero and is non-zero if at least one of the node cheats. Such *inconsistencies* in time difference estimates can severely jeopardize the accuracy of the associated time synchronization protocol. The first step towards achieving accurate time synchronization is to efficiently eliminate such inconsistent time offset data introduced by the cheating behavior of nodes participating in the time synchronization protocol.

In this paper, we study the problem of eliminating inconsistent time offset data in time synchronization protocols as a graph-theoretic optimization problem. We first formulate the time synchronization problem in sensor networks as a constraint satisfaction problem in a special type of graph of the network called the *Time Difference Graphs (TDG)*. In a time difference graph, each sensor node is a vertex and an edge exist between two nodes if they are in direct communication range of each other (one hop neighbors). Each vertex has a weight depending on the local time at the corresponding node and the weight of each directed edge is the time offset (difference) between the connecting nodes (vertices). We first prove that the time synchronization problem in the network has a solution if and only if the associated time difference graph is consistent. A time difference graph containing inconsistent time offset values is also referred to as a partially consistent time difference graph. The problem of eliminating inconsistent time offset values in time synchronization protocols then reduces to obtaining a largest consistent subgraph of the corresponding time difference graph of the network. This optimization problem is also referred here as the Maximum Consistent Time Difference Graph or the MCTD problem. We show the hardness of the MCTD problem using an efficient reduction from a well known *NP*-complete problem, i.e., the VERTEX-COVER problem and show that it is unlikely to have a polynomial time algorithm for the MCTD problem in the general case. We then study the MCTD problem for a special type of time difference graphs, namely the completely connected time difference graphs. Although the MCTD problem is also hard for this case, there exists polynomial time approximation algorithms for the MCTD problem in complete time difference graphs. We discuss two such algorithms, namely the greedy algorithm based on a greedy selection heuristic and a LP-relaxation based algorithm for an Integer Programming formulation of the above problem. However for the solution quality of the algorithms, we prove that there is no algorithm that can approximate the MCTD problem for complete time difference graphs within a factor $n^{1-\varepsilon}$, where $\varepsilon > 0$, unless $P = NP$.

1.1 Background and Related Work

The problem of time synchronization in the presence of malicious motes was first studied by Ganeriwal et al. [11]. They identify malicious attacks on existing time synchronization schemes and provide secure single-hop, multi-hop and group synchronization protocols. A major shortcoming in the algorithms proposed by them were that they simply aborted when the computed time difference was greater than some precomputed maximum expected difference. Moreover, it was not specified how this maximum expected difference was computed. Song et al. [29] try to overcome the shortcomings of the previous results by proposing two approaches to detect and accommodate a specific type of attack called the delay attack where a malicious attacker deliberately delays the transmission of synchronization messages to magnify the time offset. They proposed schemes that make use of the fact that if there is no malicious mote, the time offsets among the sensor motes should follow the same (or similar) distribution. For the attacks to be effective, malicious motes typically report their time offsets much larger than those from the benign motes, leaving their reported values suspicious. The methods proposed by Song et al. performs reasonably better, but suffers from issues like extra reference mote requirement and threshold computation that can be viewed as an overhead. Recently, Li et al. [18] propose a secure time synchronization protocol that combines the Sender-Receiver model and Receiver-Receiver model to verify the synchronization process between each synchronizing pair. The idea behind their scheme is to use a node's neighbors as verifiers to check if the synchronization is processed correctly so that it can detect the attacks launched by compromised (or malicious) nodes. The drawback of their scheme is that they assume a hierarchical node structure and presence of a set of secure verifiers for each node. The main issue is that none of the schemes discussed above formally treat the fundamental problem itself which is given a set of nodes, their local times and the time differences between nodes, is it even possible for any polynomial-time algorithm to efficiently (or optimally) pick out the trouble causing motes? Unless this question is answered, it is not possible to gauge the effectiveness of secure time synchronization schemes working under specific assumptions and conditions, like the ones discussed above.

1.2 Paper Organization

In Section 2, we outline the network and adversary model, introduce the notion of time difference graphs and define the problem of time synchronization as a constraint satisfaction problem in time difference graphs. In Section 3 we introduce the MCTD problem for general graphs and present hardness results for the same. In

Section 4, we analyze the MCTD problem for completely connected time difference graphs and present two approximation algorithms for it. In section 5, we conclude with a summary of important results and provide directions for future research.

2 Mathematical Formulation

In this section we present a graph-theoretic formulation for the time synchronization problem in sensor networks.

2.1 Network Model

Let $N = \{1, 2, \ldots, n\}$ be the set of n motes in the network. let $P = \{p_1, p_2, \ldots, p_n\}$ be the *local time vector* such that each p_i gives the local time on the mote i. We assume that this notion of local time ($p_i \in \mathbb{R}$) is nothing but the time difference in seconds from some global time scale like UTC (Universal Coordinated Time). It can be converted easily into 24/12 hour format using suitable time management software on the motes. At the start of the application when the network is deployed, $p_i = p_j \; \forall i, j \in N$, but this equality ceases to hold with time due to clock drift, clock skew and other factors as described in [30, 22]. A *Time Difference Graph (TDG)*, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$ for the network can be defined as follows. The set $\mathbb{V} = \{v_1, v_2, \ldots, v_n\}$ of vertices contains a vertex corresponding to each mote in the network. Each vertex v_i is associated with a weight $w_i = p_i$, i.e., the weight of each vertex equals to the local time on the corresponding mote. Formally, the graph $\mathbb{G}$ is associated with a *time function* w, $w : \mathbb{V} \rightarrow \mathbb{R}$, such that w assigns a weight to each vertex in the graph that signifies the value of the local time on the corresponding mote represented by that vertex. An edge exists between two vertices v_i and v_j in the graph $\mathbb{G}$ if and only if i and j are neighbors of each other. By neighbors we mean that both i and j are in the radio range of each other and can communicate with each other directly, i.e, i can send a packet to j and j can send a packet to i. They may or may not be physically close to each other. In each such pair i, j, one node is a *sender* node and one node is a *receiver* node. For each neighbor pairs i, j, the receiver node computes its time difference with respect to the sender node and their is a *directed* edge $(v_i, v_j) \in \mathbb{E}$ from v_i to v_j in $\mathbb{G}$. The weight δ_{v_i,v_j} of each edge $(v_i, v_j) \in E$ is the estimated time offset (difference) computed by the receiver from the sender. More formally, the graph $\mathbb{G}$ is associated with a *time difference function* δ, $\delta : \mathbb{E} \rightarrow \mathbb{R}$, such that δ assigns a weight to each edge in the graph signifying the *estimated* value of the time difference between the two vertices (motes) connected

by that directed edge. δ_{v_i,v_j} is positive if the receiver node leads the sender node, δ_{v_i,v_j} is negative if the receiver node lags the sender node and δ_{v_i,v_j} is zero if there is no time difference between the receiver and sender node. An important thing to note here is that δ_{v_i,v_j} may not necessarily be equal to $w_i - w_j$, i.e., the difference of the corresponding local clocks. We will see later that $\delta_{v_i,v_j} = w_i - w_j$ only if both the nodes honestly participate in the time difference estimation. Moreover, we can safely assume that the time difference function can be efficiently computed. An example of one such efficient implementation of the time difference function is given by Ganeriwal et al. [12] where the receiver computes the time difference by receiving time stamped messages from the sender. The set $\mathbb{E}$ in $\mathbb{G}$ is the set of all the edges as defined above. To simplify the current exposition we assume here that the graph $\mathbb{G}$ is a simple, planar, connected graph i.e., every vertex is reachable from every other vertex through a sequence of edges (or there are no disconnected vertices), no edges cross each other and there are no self loops. But, we will see later in Section 3 that the results are general enough and can be applied to incomplete (or disconnected) graphs also. A TDG with 6 nodes (vertices) and 8 directed edges is depicted in Fig. 2.

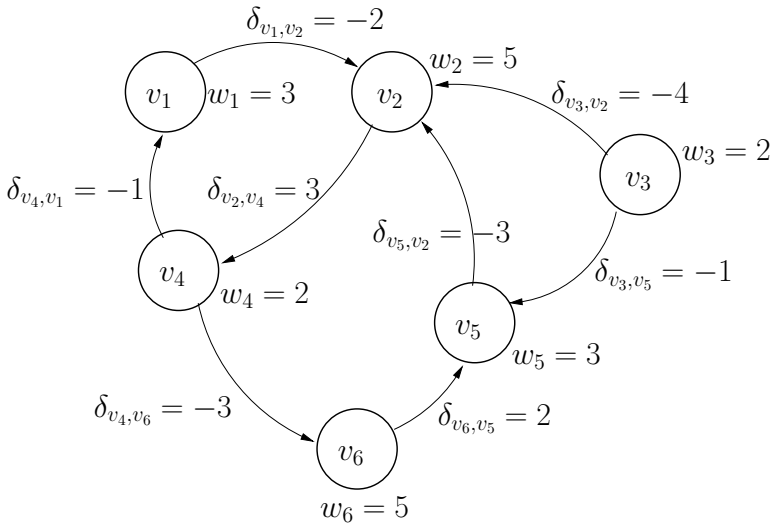

Figure 2: Time Difference Graph (TDG)

2.2 The Time Synchronization Problem

Given a TDG $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$ for the network, where w is the time function and δ is the time difference function for $\mathbb{G}$, the problem of *time synchronization* in the network can be represented as a *Constraint Satisfaction Problem* [27]. We define a set of $\|\mathbb{V}\| = n$ variables $x_1, x_2, \ldots, x_n$, one for each vertex $v_i \in \mathbb{V}$

in $\mathbb{G}$. These variables are also called the *adjustment variables*. Each adjustment variable x_i has a non-empty domain, which in this case is the set of real numbers $\mathbb{R}$. Each directed edge $(v_i, v_j) \in \mathbb{E}$ defines two constraints. The first constraint, denoted as $C_{1(v_i,v_j)}$, gives the relationship between x_i, x_j and the time difference function δ and is given as

$$C_{1(v_i,v_j)} \equiv x_j - x_i = \delta(v_i, v_j) \tag{1}$$

The second constraint, denoted as $C_{2(v_i,v_j)}$, gives the relationship between x_i, x_j and the time function w and is given as

$$C_{2(v_i,v_j)} \equiv w_i + x_i = w_j + x_j \tag{2}$$

Thus, there are a total of $2\|E\|$ constraints in the system. The state of a Constraint Satisfaction Problem (CSP) is defined by an *assignment* of values to some or all of the variables, $\{x_1 = m_1, x_2 = m_2, \ldots, x_n = m_n\}$. An assignment that does not violate any constraints is called a *consistent* or *legal* assignment. A *complete assignment* is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints. Given a TDG, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$, for the network, the problem of time synchronization is then determining a complete consistent assignment to the adjustment variables $x_1, x_2, \ldots, x_n$, i.e., an assignment that includes all adjustment variables $x_1, x_2, \ldots, x_n$ and satisfies all the constraints in the system. Time synchronization is *feasible* or *solvable* in the network if such a complete assignment exists. If such a complete assignment does not exist then it implies that some of the constraints in the network cannot be satisfied and time synchronization is infeasible or partially feasible. This infeasibility may be due to the cheating behavior by some of the nodes and is outlined in detail in the following section.

2.3 Adversary Model and Inconsistency in Time Synchronization

From the formulation above, we can see that a solution to the time synchronization problem is any assignment to the adjustment variables $\{x_1 = m_1, x_2 = m_2, \ldots, x_n = m_n\}$ that satisfies all the $2\|E\|$ constraints in the time difference graph. But, all the constraints may not be satisfied by any assignment if some (or all) of the constraints are not consistent with each other. To keep the exposition simple, we currently assume that there is no measurement error. But, we will see later on that the scheme and the problem formulation is general enough and can be easily extended to practical systems by adding a small measurement error constant. We

assume that if all nodes behave honestly then the local times are reported correctly and time differences estimated accurately. On the other hand, when the nodes fail to honestly participate in the time synchronization protocol the reported local times and time difference estimations are arbitrary. This cheating behavior is reflected in the time function w and the time difference function δ of the corresponding time difference graph of the network. In reality, nodes in time synchronization protocols can cheat in the following ways,

1. Advertising incorrect local clock: Each node i advertises the value of its local clock p_i to the neighboring nodes during the time synchronization protocol. A node can cheat by advertising incorrect local time information to other nodes. This cheating behavior translates to the time function w in the time difference graph model assigning incorrect w_i values to the corresponding vertices v_i in the TDG $\mathbb{G}$ of the network. Advertising incorrect local time by a sender node i may also affect the time difference computation by the receiver node, say j. This translates to the time difference function δ assigning incorrect weights δ_{v_i,v_j} to the directed edges (v_i, v_j) in the time difference graph model.

2. Manipulating message delay or time-stamp information: During the time synchronization protocol, a cheating node i (either sender or receiver) can manipulate message transmissions by introducing unnecessary delays or changing packet time stamps. This also affects the time difference computation and the time difference function δ in the time difference graph model.

There is no globally verifiable way to detect if a node v_i is cheating about its local time value w_i (point 1) or if the nodes local clock is actually way out of sync with the other nodes. Cheating on ones local time value may or may not lead to incorrect computation of time difference values by the neighboring (receiver) nodes on edges out of the cheating node. A node can obviously choose not to cheat on its local time value and still manipulate communications (as discussed in point 2 above) with other nodes in order to adversely affect time difference computations. But, there is a globally verifiable way to detect if the time difference estimations are correctly computed or not, although it may not be possible to pin-point which node is cheating in case the verification fails. This verification, called the triangle Law, was briefly introduced in Section 1. Thus, we would like to reiterate that the time difference function (δ) does not directly depend on the time function w but depends on the behavior (cheating or honest) of the sender node and that cheating behavior can be detected by verifying the time difference estimates using the triangle law. Before we move ahead, we need some definitions.

Definition 2.1. *Cycle or Circuit: A cycle in a graph is an alternating sequence of vertices and edges in a*

graph, with each edge being incident to the vertices immediately preceding and succeeding it in the sequence such that all the vertices in the sequence are distinct except the first and the last.

In this paper, the usage of word cycle implicitly always implies a simple cycle, i.e. a cycle with no repeated vertices except the first and the last vertex. Recollect that the triangle law for time offset (time difference) gives the necessary condition for time difference consistency for a group of 3 nodes. Here, we present a more general notion of a similar concept called the *consistent cyle*.

Definition 2.2. *Consistent Cycle: Given a TDG,* $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$*, any cycle of* $\mathbb{G}$ *consisting of 3 or more vertices is called a consistent cycle if and only if the sum of the time difference function values* δ_{v_i,v_j} *of all the edges* (v_i, v_j) *in the cycle is exactly zero.*

A cycle of 3 or more vertices in which the sum of the time difference function values for all the edges is anything except zero (positive or negative) is called an inconsistent cycle. A time difference graph that contains no inconsistent cycles is called a *consistent* time difference graph. A time difference graph that contains inconsistent cycles is called a *inconsistent* or *partially consistent* time difference graph. One problem with the current definition of time difference graphs is the presence of *connected acyclic loops*, which is possible in directed graphs. There is no way to check for the consistency of such a loop in the existing definition of the time difference graph. We overcome this problem by adding redundancy as discussed in the next section. A time difference graph which does not have any connected acyclic loops and which does not any other directed cycles is always consistent.

2.4 Time Difference Graphs - Revisited

In a directed graph like the time difference graph (as discussed in Section 2.1), it is possible that some loops or vertex combinations may be connected by edges but may not form a cycle (connected acyclic loops). An example of such an acyclic loop in the time difference graph shown in Fig. 2 is $< v_2, v_3, v_5 >$. As a result, it is not possible to apply the triangle law to such acyclic loops and check for their consistency and thus the overall consistency of the graph. In order to overcome this problem, we add some redundancy to the definition of time difference graphs. Given a TDG, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$, we define a new graph $\mathbb{G}' = (\mathbb{V}', \mathbb{E}', w', \delta')$ as follows. The vertex set and time difference function of $\mathbb{G}'$ is the same as that of $\mathbb{G}$, i.e., $\mathbb{V} = \mathbb{V}'$ and $w = w'$. For each edge $(v_i, v_j) \in E$, $(v_i, v_j) \in E'$ and $(v_j, v_i) \in E'$. Also if $(v_i, v_j) \in E$ and $\delta(v_i, v_j) \neq 0$ then $\delta'(v_i, v_j) = \delta(v_i, v_j)$ and $\delta'(v_j, v_i) = -1 \times \delta(v_i, v_j)$. If $\delta(v_i, v_j) = 0$ then $\delta'(v_i, v_j) = \delta'(v_j, v_i) = 0$. This

new graph is called the *Redundant Time Difference Graph (RTDG)* and is shown in Fig. 5. As we can see

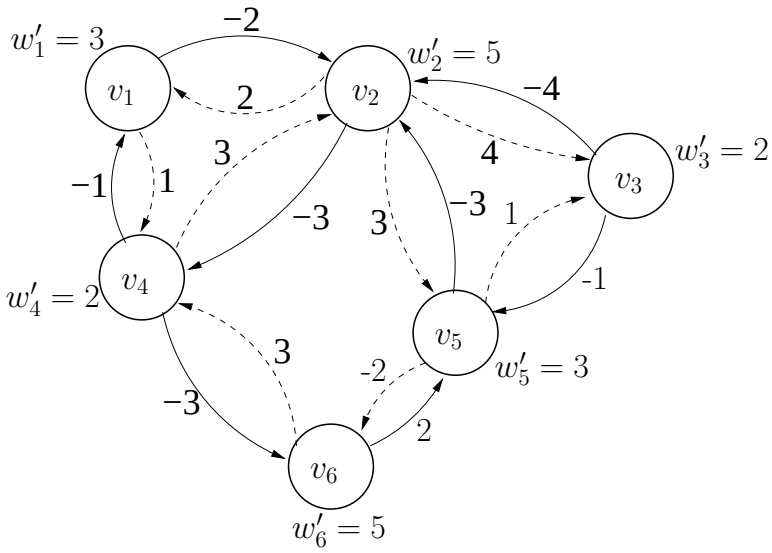

Figure 3: Redundant Time Difference Graph (RTDG)

from Figure 5, for each edge in the time difference graph, a new edge between the same pair of nodes in the opposite direction is added in the corresponding redundant time difference graph. This edge is called the *redundant edge* as its time difference value is just opposite in sign to the time difference value of the actual edge and it does not provide any new information. This is also intuitive because if a node, say *A*, lags another node, say *B*, by *m*, then the same thing can also be interpreted as *B* leads *A* by *m*. Redundant time difference graphs eliminate acyclic loops in time difference graphs by converting each such loop into a directed cycle which can be verified for consistency. This brings us to our first result which gives the relationship between the solution of the time synchronization problem in a network and the consistency of the corresponding redundant time difference graph.

Theorem 2.1. *The time synchronization problem for a redundant time difference graph* $\mathbb{G}' = (\mathbb{V}', \mathbb{E}', w', \delta')$ *has a solution if and only if* $\mathbb{G}'$ *is consistent.*

Proof. Let, $\mathbb{G}' = (\mathbb{V}', \mathbb{E}', w', \delta')$ be a redundant time difference graph as defined in Section 2.4. We first prove the reverse direction. We will show that if the graph $\mathbb{G}'$ is consistent then the time synchronization problem for $\mathbb{G}'$ has a solution. In other words, if the graph $\mathbb{G}'$ is consistent then the CSP for the time synchronization problem has at least one assignment to the adjustment variables $x_1, x_2, \ldots, x_n$ such that all the constraints (Eqn. 1 and 2) are satisfied. We prove this by a contradiction argument. Since the graph $\mathbb{G}'$

is consistent, by definition of consistency all simple cycles in $\mathbb{G}'$ are consistent, i.e., sum of time difference values of all edges in each cycle is zero. Now, let $x_1 = m_1, x_2 = m_2, \ldots, x_n = m_n$ be one assignment to the adjustment variables such that there is at least one constraint that is not satisfied. Let this constraint (which is not satisfied) be on the edge (v_i, v_{i+1}), i.e., $x_{i+1} - x_i \neq \delta(v_i, v_{i+1})$. Thus, $\delta(v_i, v_{i+1}) \neq m_{i+1} - m_i$. Without loss of generality, assume that $\delta(v_i, v_{i+1}) < m_{i+1} - m_i$. Also let this edge be on some cycle $\mathbb{C}' = (v_1, v_2), (v_2, v_3), \ldots, (v_{i-1}, v_i), (v_i, v_{i+1}), (v_{i+1}, v_{i+2}), \ldots (v_r, v_i)$. Now, since $\mathbb{G}'$ is consistent, the cycle $\mathbb{C}'$ is consistent. Thus,

$$
\begin{aligned}
& \delta(v_1, v_2) + \delta(v_2, v_3) + \ldots + \delta(v_{i-1}, v_i) + \delta(v_i, v_{i+1}) + \delta(v_{i+1}, v_{i+2}) + \ldots + \delta(v_{r-1}, v_r) = 0 \\
\Longrightarrow \quad & (x_2 - x_1) + (x_3 - x_2) + \ldots + (x_i - x_{i-1}) + \delta(v_i, v_{i+1}) + (x_{i+2} - x_{i+1}) + \ldots + (x_1 - x_r) = 0 \\
\Longrightarrow \quad & (m_2 - m_1) + (m_3 - m_2) + \ldots + (m_i - m_{i-1}) + \delta(v_i, v_{i+1}) + (m_{i+2} - m_{i+1}) + \ldots + (m_1 - m_r) = 0 \\
\Longrightarrow \quad & m_i - m_{i+1} + \delta(v_i, v_{i+1}) = 0 \\
\Longrightarrow \quad & \delta(v_i, v_{i+1}) = m_{i+1} - m_i
\end{aligned}
$$

which is a contradiction. This proves the reverse direction. Similarly, the forward direction is also straightforward. □

We would like the readers to note that from this point onwards, the term time difference graph would always imply a redundant time difference graph unless explicitly specified. Next, we discuss the problem of efficiently eliminating inconsistencies in time difference graph models for sensor network systems.

3 Eliminating Inconsistencies in Time Difference Graphs

Up to this point, we have formulated the distributed time synchronization problem in sensor network systems as a constraint satisfaction problem in a graph-based model of the system, namely time difference graphs. We also proved the fundamental result that a feasible solution to the time synchronization problem exists if and only if the corresponding time difference graph is consistent. This brings us to the main issue which is given a partially consistent time difference graph, how to obtain the largest consistent subgraph of this graph? This is equivalent to the problem of efficiently eliminating inconsistent (or cheating) behavior from such graph-based representations for time synchronization protocols. This problem is formally stated next.

3.1 Maximum Consistent Time Difference Graph (MCTD) Problem

A consistent subgraph of a partially consistent time difference graph is obtained by eliminating vertices (and the corresponding directed edges incident on and coming out of these vertices) until the resulting induced subgraph is consistent, i.e., all simple cycles are consistent. The *size* of the consistent subgraph is the cardinality of its vertex set. The *edge size* is the cardinality of its edge set. A consistent subgraph is *maximal* if its vertex set is not a proper subset of the vertex set of any other consistent subgraph of that time difference graph. A *maximum* consistent subgraph is a maximal consistent subgraph with maximum size. Now, given a time difference graph, the problem of obtaining the largest consistent subgraph can be formulated as an optimization problem as follows: Given a partially consistent TDG, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$, find the maximum consistent subgraph of $\mathbb{G}$. We refer to this problem as the *Maximum Consistent Time Difference Graph Problem* or MCTD. A decision (or parameterized) version of the problem can be stated as,

MCTD

Input: A partially consistent time difference graph $\mathbb{G} = (\mathbb{V}, \mathbb{E}, w, \delta)$ and a positive integer k s.t. $k \leq \|V\|$.

Question: Does $\mathbb{G}$ contain a consistent time difference subgraph of size k or more?

3.2 Hardness of the MCTD Problem

Intuitively, the MCTD problem appears to be hard. In reality, the MCTD problem does belongs to the class of highly intractable problems. We do not have sufficient proof that it even belongs to NP, i.e., the class of non-deterministic polynomial time algorithms. This is because it is highly unlikely that MCTD has a *polynomial time verifier*. Given a TDG, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \delta)$[‡] and an integer $k \leq \|\mathbb{V}\|$, it is not possible to verify in polynomial time whether a subgraph $\mathbb{G}'$ of $\mathbb{G}$ (of size k) contains only consistent cycles. To verify the consistency of a graph, all the simple cycles need to be verified for consistency. The total number of cycles in $\mathbb{G}'$ itself could be exponential. But, we do show that MCTD is *NP-hard*, i.e., it is at least as hard as every problem in NP. We prove this result by a straightforward polynomial time many-one (hardness preserving) reduction from a well-known *NP*-complete problem, the VERTEX-COVER problem [13, 15]. A vertex cover of a directed graph $\tilde{G} = (\tilde{V}, \tilde{E})$ is a subset of vertices $C \subseteq \tilde{V}$ that contains at least one vertex of every directed edge $(\tilde{u}, \tilde{v}) \in \tilde{E}$, and the VERTEX-COVER problem (also called MINIMUM VERTEX COVER problem) is to find such a subset C of the smallest cardinality.

[‡]We ignore w in the following exposition since it is not used to check consistency of the time difference graph

Theorem 3.1. *The Maximum Consistent Time Difference Graph (MCTD) problem is NP-hard.*

Proof. We show that VERTEX-COVER $\leq_m^P$ MCTD, i.e., VERTEX-COVER many-one (m) reduces in polynomial time (P) to the MCTD problem.

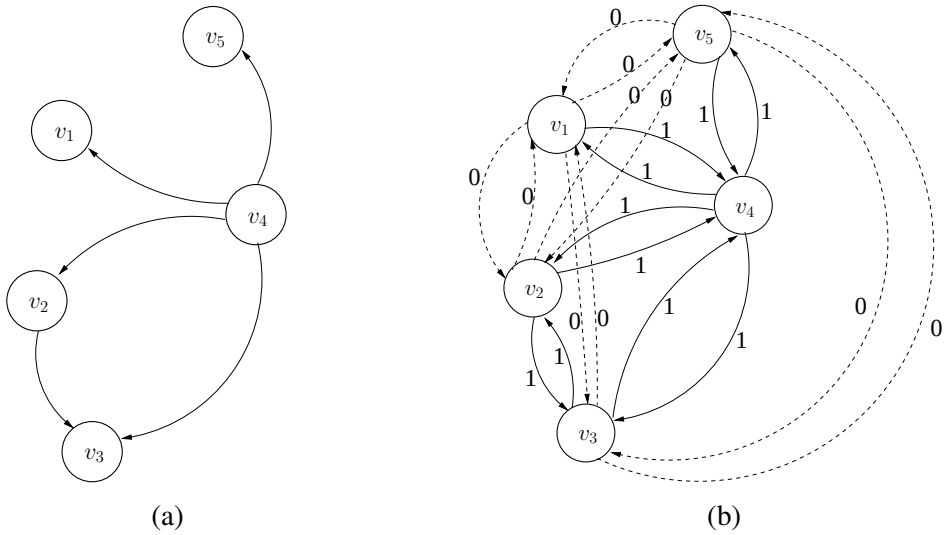

Figure 4: (a) Input graph for the VERTEX - COVER problem, $\tilde{G} = (\tilde{V}, \tilde{E})$; (b) Input graph for the MCTD problem, $\mathbb{G} = (\mathbb{V}, \mathbb{E})$

Construction: We describe a polynomial time construction that maps an instance $\tilde{G} = (\tilde{V}, \tilde{E})$ of the VERTEX-COVER problem to an instance $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \delta)$ of the MCTD problem such that $\tilde{G}$ has a vertex cover of size $\leq k$ ($k \leq \|\tilde{V}\|$) if and only if $\mathbb{G}$ has a consistent subgraph of size $\geq \|\tilde{V}\| - k$. Since, consistency is defined in terms of the time difference function δ only, we can omit the time function w in the current exposition. The construction is shown in Figure 4.

1. For each vertex v in the vertex set $\tilde{V}$ of $\tilde{G}$, place a vertex v in the vertex set $\mathbb{V}$ of $\mathbb{G}$.

2. For each directed edge $(u,v) \in \tilde{E}$, add an edge (u,v) in $\mathbb{E}$ of $\mathbb{G}$. If $(v,u) \notin \tilde{E}$, add an edge (v,u) in $\mathbb{E}$ of $\mathbb{G}$. For each such edge, define $\delta(u,v) = 1$ and $\delta(v,u) = 1$. These edges are shown as solid lines in Figure 4(b).

3. For each vertex pair $u, v \in V$ such that edge $(u,v) \notin \tilde{E}$ and $(v,u) \notin \tilde{E}$, add an edge (u,v) and (v,u) in $\mathbb{E}$ of $\mathbb{G}$. For each such edge, define $\delta(u,v) = 0$ and $\delta(v,u) = 0$. These edges are shown as dotted lines in Figure 4(b).

It is clear that the above construction can be completed in polynomial time. We now show that the graph $\tilde{G}$ has a vertex cover of size k if and only if the graph $\mathbb{G}$ has a consistent time difference graph of size $\|\tilde{V}\| - k$.

Suppose the graph $\tilde{G}$ in Figure 4 has a vertex cover C ($C \subseteq \tilde{V}$) of size k ($\|C\| = k$). Since C is a vertex cover, $\forall (u,v) \in \tilde{E}$, either u or v or both are in C. By our construction, $\forall (u,v) \in \tilde{E}$, $(u,v) \in \mathbb{E}$ and $(v,u) \in \mathbb{E}$. Thus, C also covers all the edges with time difference values 1 in the time difference graph $\mathbb{G}$. This implies, $\mathbb{V} - C$ would contain no edges with time difference values 1, i.e., $\mathbb{V} - C$ would contain no cycle such that the sum of the time difference values of all the edges in the cycle is not equal to 0. In other words, subgraph induced by $\mathbb{V} - C$ is a consistent time difference graph of $\mathbb{G}$ of size $\|\mathbb{V}\| - k$. Thus, if any directed graph $\tilde{G}$ has a vertex cover of size k, the time difference graph $\mathbb{G}$ has a consistent subgraph of size $\|V\| - k$.

Now we prove the other direction. First thing we observe in the above construction is that the time difference graph $\mathbb{G}$ is complete, i.e., there is an edge from every vertex to every other vertex. Moreover, there are edges of the same type in both direction from every vertex to every other vertex. This implies that any induced subgraph of $\mathbb{G}$ is also complete. Let C' be the vertex set representing the consistent time difference subgraph of $\mathbb{G}$ of size m ($m \leq \|\mathbb{V}\|$). Thus, C' is a complete graph with no cycles with sum of time difference values of the edges (in the cycle) not equal to zero. This implies that there can be no edge in C' with time difference value 1, otherwise there will be at least one inconsistent cycle in C'. Thus, $\mathbb{V} - C'$ covers all edges in $\mathbb{E}$ with time difference value 1. From our construction, it is clear that the edge set $\tilde{E}$ of $\tilde{G}$ is a subset of all edges in $\mathbb{E}$ with time difference value 1. Thus, $\mathbb{V} - C'$ covers all the edges in $\tilde{G}$ and is a vertex cover of $\tilde{G}$ of size $\|\mathbb{V}\| - m$ i.e., $\|\tilde{V}\| - m$ since $\|\tilde{V}\| = \|\mathbb{V}\|$

Thus, VERTEX-COVER many-one reduces in polynomial time to MCTD. Since VERTEX-COVER is *NP*-complete, MCTD is *NP*-hard. □

Thus, it is very unlikely that MCTD will have a deterministic polynomial time algorithm unless $P = NP$. Moreover, it seems unlikely that MCTD will even have an efficient approximation algorithm.

3.3 Solving the MCTD Problem

In this section, we present a very simple technique for obtaining a maximum consistent subgraph, given a TDG, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \delta)$. But before we do that, we need to introduce another graph-based problem, namely FEEDBACK VERTEX SET problem [13]. The FEEDBACK VERTEX SET problem is defined as: given a directed or undirected graph, $G = (V,E)$, find a subset $F \subseteq V$ of vertices in the graph such that $G - F$ is acyclic. In simpler words, the FEEDBACK VERTEX SET problem is to find a (minimum) subset of vertices that covers all the cycles in G. F is called the feedback vertex set of G, or an FVS of G. Then, the algorithm

for solving MCTD can be given as shown in Algorithm 1,

1: Compute the set of all the negative cycles C in the graph $\mathbb{G}$. A negative cycle is a cycle such that the sum of the weights of the edges in the cycle is < 0.
2: Compute the feedback vertex cover F of C.
3: **return** $\mathbb{G} - F$ as the maximum consistent time difference graph of $\mathbb{G}$

Algorithm 1: Calculating Maximum Consistent Subgraph for the TDG, $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \delta)$ (Section 2.4)

Thus, the MCTD problem for a time difference graph can be solved, if efficient algorithms exists for finding all the negative cycles of a weighted directed graph and for solving the FEEDBACK VERTEX SET problem for directed graphs. But, both the above problems are known *NP*-complete problems. Although, deciding the existence and finding a negative cycle in a weighted directed graph is polynomially solvable and all cycles of a directed or undirected graph can be enumerated efficiently by a simple backtracking algorithm [25], Kachiyan et al. [16] proved that (directed) negative cycles in a graph cannot be generated in polynomial output time, unless $P = NP$. Using Gallai's results [10], Kachiyan et al. showed that the hard problem of finding minimal infeasible subsystems of a system of linear inequalities, called IIS (Irreducible Inconsistent Subsystems) corresponds in a one-to-one way to the problem of finding the maximum number of negative cycles in an edge-weighted directed graph. Similarly, Karp et al. [15] proved the *NP*-completeness of the FEEDBACK VERTEX SET problem. For undirected graphs, there are considerable results for the FEEDBACK VERTEX SET problem, e.g., there are exact algorithms of finding a minimum FVS in a graph on n vertices in time $O(1.9053^n)$ [24] and in time $O(1.7548^n)$ [9]. There is also a polynomial time 2-approximation algorithm for it [4]. In directed graphs, the FEEDBACK VERTEX SET problem becomes harder and there has been only a limited progress on it, since Karp proved that to find an FVS in a directed graph of size bounded by k is NP-complete [15]. No exact algorithms with running time within $O(c^n n^{O(1)})$, where $c < 2$, and no polynomial time approximation algorithms with constant ratio have been found [8]. Thus, from the above results it appears unlikely that the MCTD problem will have a polynomial time (in terms of the number of vertices) exact algorithm for the general time difference graphs. It is still an open question if a constant ratio approximation algorithm exist for the MCTD problem. In the next section, we discuss an interesting generalization of the MCTD problem discussed above. We analyze the MCTD problem for a special type of time difference graph, namely a completely connected time difference graph.

4 MCTD Problem for Completely Connected TDG

A *completely connected (or complete)* time difference graph is a special type of redundant time difference graph in which there is an (directed) edge between every pair of vertices in the graph (in both the directions). Although, it is very difficult to model existing sensor networks (except extremely dense network spread over a small area) using such completely connected graphs, they possess certain very interesting structural properties. This motivates us to further investigate whether the MCTD problem is equally hard for such completely connected graphs or if efficient solutions exist for this special case.

4.1 Properties of Completely Connected TDG

Property 1. *A complete time difference graph has a polynomial number of exactly three node (vertex) cycles. Specifically, there are a total of* $2.\begin{pmatrix} n \\ 3 \end{pmatrix}$*, where n is the total number of vertices in the graph.*

Property 2. *A complete time difference graph is consistent if and only if all the three node (vertex) cycles in the graph are consistent.*

Property 1 is obvious from the structure of the complete time difference graph. Property 2 follows from Lemma 4.1.

Lemma 4.1. *In a complete time difference graph, two 3-node cycles with two common vertices are independently consistent if and only if the cycle containing all the vertices (of the two cycles) is consistent.*

Proof. We give the proof for this Lemma by a contradiction argument. We first prove the forward direction,

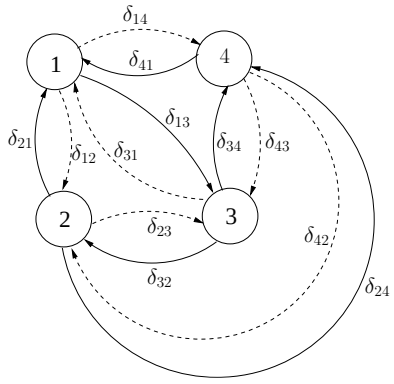

Figure 5: Property 2 of Completely Connected TDG

i.e., if two 3-node cycles with two common vertices are independently consistent then the cycle containing

all the vertices (of the two cycles) is consistent. Without loss of generality let us assume that {1,2,3,1} and {1,3,4,1} are the two 3-node independently consistent cycles with the common nodes 1 and 3. Thus, $\delta_{12}+\delta_{23}+\delta_{31}=0$ and $\delta_{13}+\delta_{34}+\delta_{41}=0$. Also, let {1,2,3,4,1} be the cycle containing all the four nodes of the above two 3-node cycles and let cycle $\{1,2,3,4,1\}$ be not consistent. Thus,

$$\begin{aligned}
& \delta_{12}+\delta_{23}+\delta_{34}+\delta_{41}\neq 0 \\
\Longrightarrow \quad & \delta_{12}+\delta_{23}+\delta_{31}-\delta_{31}+\delta_{34}+\delta_{41}\neq 0 \\
\Longrightarrow \quad & -\delta_{31}+\delta_{34}+\delta_{41}\neq 0 \quad (\delta_{12}+\delta_{23}+\delta_{31}=0) \\
\Longrightarrow \quad & \delta_{13}+\delta_{34}+\delta_{41}\neq 0 \quad (\delta_{13}=-\delta_{31}) \\
\Longrightarrow \quad & \textit{Cycle } \{1,3,4,1\} \textit{ is inconsistent}
\end{aligned}$$

which is a contradiction. Similarly the reverse direction can be proved by a similar argument. □

Next, we outline some important theoretical results for the MCTD problem in completely connected time difference graphs.

4.2 Theoretical Results

Contrary to our initial intuition that there might be a possibility of better results for the MCTD problem in complete time difference graphs, theoretical analysis has shown that the MCTD problem is indeed hard. Our next result proves that the MCTD problem for complete time difference graphs is *NP*-complete.

Theorem 4.1. *MCTD problem for completely connected time difference graph is NP-complete.*

This result follows from a very straight-forward polynomial time reduction from a well-known *NP*-hard problem, called the MAX-CLIQUE problem [15]. The MAX-CLIQUE problem for a given graph with n vertices is to find a maximum size clique in it, i.e., a complete subgraph of maximum size. The best known algorithm for approximating MAX-CLIQUE has a factor of $O(\frac{n}{log^2 n})$ [5]. Moreover, it has been proved that for any $\varepsilon > 0$ there is no polynomial time approximation algorithm for CLIQUE within factor $n^{1-\varepsilon}$, unless $P = NP$ [14]. As a lemma to the above theorem, we show that solving the MCTD problem for completely connected time difference graph is at least as hard as solving the MAX-CLIQUE problem for a graph.

Lemma 4.2. *MCTD problem for completely connected time difference graph is hard to approximate within a factor $n^{1-\varepsilon}$ ($\varepsilon > 0$), unless P=NP.*

Due to space constraints, we do not include the proofs for Theorem 4.1 and Lemma 4.2 here. From Lemma 4.2, it follows that the MCTD problem for completely connected time difference graphs will have no approximation algorithm within a factor $n^{1-\varepsilon}$ unless $P = NP$, where n is the number of vertices in the graph and for some constant $\varepsilon > 0$.

4.3 Algorithms

We now propose two approximation algorithms for the MCTD problem in complete time difference graphs. The first algorithm is based on a greedy strategy for selecting vertices from the complete time difference graph. The second algorithm is based on solving the LP relaxation of an Integer-Programming formulation for the MCTD problem.

4.3.1 Greedy Strategy

This algorithm is based on a greedy heuristic for selecting and adding vertices into the set of consistent time difference subgraph for a complete time difference graph. This selection is based on the *Consistency Index (CI)* of a vertex.

Definition 4.1. *Consistency Index (CI): The consistency index of a vertex in a complete time difference graph is the total number of consistent 3-node cycles of the graph that the vertex is present in.*

The greedy algorithm for the MCTD problem is shown in Algorithm 2,

Require: Time Difference Graph $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \delta)$, $\|\mathbb{V}\| = n$.
1: Compute CI for each vertex, $v_i \in \mathbb{V}$.
2: Sort vertices based on CI from largest to smallest. Let the sorted list be $\mathbb{V}' = \{v_1, v_2, \ldots, v_n\}$.
3: Let the resulting MCTD subgraph for $\mathbb{G}$ be represented as $\mathbb{T}$. Initially, let $\mathbb{T} = v_1 \in \mathbb{V}'$.
4: **for** each vertex $v_i \in \mathbb{V}'$ ($i = 2$ to n) **do**
5: **if** v_i is consistent with every vertex in $\mathbb{T}$ (i.e., all 3-vertex cycles in $\mathbb{T}$ after adding v_i is consistent) **then**
6: Add v_i to $\mathbb{T}$.
7: **end if**
8: **end for**
9: **return** $\mathbb{T}$.

Algorithm 2: Greedy Strategy for MCTD Problem in Complete TDG

The CI computation step of the greedy algorithm takes $O(n^4)$ time. This is because there are $O(n^3)$ 3-vertex cycles in the complete time difference graph and it takes another $O(n)$ time to assign a CI for every vertex in the graph. Sorting takes $O(n.logn)$ and the verification and addition steps (4 through 8) take $O(n^4)$ in the worst case. Thus, the total running time of the greedy algorithm for the MCTD problem is $O(n^4)$, which is still polynomial in terms of the number of vertices.

4.3.2 Integer Programming Formulation

The MCTD problem for a complete time difference graph can also be formulated as an Integer Program (IP) (more specifically a 0-1 Program) as shown below:

$$
\begin{aligned}
\textit{Maximize} \quad & \sum_{i=1}^{n} x_i \\
\textit{Subject to} \quad & (x_i + x_j + x_k - 2).\delta_{i,j,k} \leq 0 \\
& \qquad \forall i,j,k \ \textit{such that} \ \ i,j,k \in \{1,2,\ldots,n\} \ \textit{and} \ \delta_{i,j,k} \equiv \delta_{i,j} + \delta_{j,k} + \delta_{k,i} \geq 0 \\
& \textit{and} \ x_i, x_j, x_k \in \{0,1\}
\end{aligned}
$$

Solving an Integer Program is a well-known NP-hard problem [15]. To overcome this problem, Linear Program (LP) relaxation for the above Integer program can be obtained. LP relaxation is solvable in polynomial time using efficient techniques like simplex algorithm. If the LP relaxation has integral solution then that can be the solution for the above IP also. But if LP relaxation does not have integral solution then techniques like rounding, branch and bound etc. can be used to obtain a feasible solution.

5 Conclusion

In this paper, we addressed the problem of eliminating cheating behavior in time synchronization protocols for highly distributed systems like wireless sensor networks. We modeled the time synchronization problem in such networks as a constraint satisfaction problem using a graph-based representation of the network, called the time difference graph. The problem of eliminating cheating (or inconsistent) behavior is then formulated as an optimization problem, called the MCTD problem, in such graphs. We proved that the MCTD problem for the general case is *NP*-hard and also analyzed the problem for special types of time difference graphs, namely complete time difference graphs. We showed that even for a complete time

difference graph, MCTD problem is *NP*-complete and there is no algorithm that can approximate it within a factor $n^{1-\varepsilon}$, $\varepsilon > 0$. Finally, we outlined two simple polynomial time approximation algorithms for the MCTD problem in complete time difference graphs.

As a future exercise we would like to answer the question: What is the best approximation ratio possible for the MCTD problem (in both the cases).

References

[1] Global clock synchronization in sensor networks. *IEEE Transactions on Computers*, 55(2):214–226, 2006. Qun Li and Daniela Rus.

[2] K. Arvind. Probabilistic clock synchronization in distributed systems. *IEEE Trans. Parallel Distrib. Syst.*, 5(5):474–487, 1994.

[3] Hagit Attiya, David Hay, and Jennifer L. Welch. Optimal clock synchronization under energy constraints in wireless ad-hoc networks. In *The 9th International Conference on Principles of Distributed Systems (OPODIS)*, pages 169–179, December 2005.

[4] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[5] R.B. Boppana and M.M. Halldórsson. *Approximating Maximum Independent Sets by Excluding Subgraphs*, volume 447, chapter Proceedings of SWAT 1990, Lecture Notes in Computer Science, pages 13–25. Springer Berlin / Heidelberg, 1990.

[6] Flaviu Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3:146–158, 1989.

[7] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Operating Systems Rev.*, 36(SI):147–163, 2002.

[8] Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating Minimum Feedback Sets and Multicuts in Directed Graphs. *Algorithmica*, 20(2):151–174, 1998.

[9] Fedor V. Fomin, Serge Gaspers, and Artem V. Pyatkin. *Finding a Minimum Feedback Vertex Set in Time $O(1.7548^n)$*, volume 4169, chapter IWPEC 2006, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006.

[10] T. Gallai. Maximum-minimum s*ä*tze *ü*ber graphen. *Acta Mathematicae, Academiae Scientiarum Hungaricae*, 9, 1958.

[11] Saurabh Ganeriwal, Srdjan Capkun, Chih-Chieh Han, and Mani B. Srivastava. Secure time synchronization service for sensor networks. In *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, pages 97–106, New York, NY, USA, 2005. ACM Press.

[12] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, New York, NY, USA, 2003. ACM Press.

[13] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 1979.

[14] J. Hastad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematics*, 182:105–142, 1999.

[15] R.M. Karp. *Complexity of Computer Computations*, chapter Reducibility Among Combinatorial Problems, pages 85–104. Plenum Press, 1972.

[16] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, and Vladimir Gurvich. Generating all vertices of a polyhedron is hard. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 758–765, 2006.

[17] Michael D. Lemmon, Joydeep Ganguly, and Lucia Xia. Model-based clock synchronization in networks with drifting clocks. In *PRDC '00: Proceedings of the 2000 Pacific Rim International Symposium on Dependable Computing*, page 177, Washington, DC, USA, 2000. IEEE Computer Society.

[18] Hui Li, Yanfei Zheng, Mi Wen, and Kefei Chen. *A Secure Time Synchronization Protocol for Sensor Network*, volume 4819, chapter Emerging Technologies in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, pages 515–526. Springer Berlin / Heidelberg, 2007.

[19] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Ĺdeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[20] David L. Mills. Internet time synchronization: The network time protocol. In *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*. 1994.

[21] Michael Mock, Reiner Frings, Edgar Nett, and Spiro Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *SRDS '00: Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, page 125, Washington, DC, USA, 2000. IEEE Computer Society.

[22] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. Technical Report UM-CS-1998-043, , 1998.

[23] S. Ping. Delay measurement time synchronization for wireless sensor networks. *Intel Research, IRB-TR-03-013*, June 2003.

[24] Igor Razgon. *Exact Computation of Maximum Induced Forest*, volume 4059, chapter Proceedings of SWAT 2006, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006.

[25] R.C. Read and R.E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5, 1975.

[26] Kay Römer. Time synchronization in ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 173–182, New York, NY, USA, 2001. ACM Press.

[27] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter Constraint Satisfaction Problems, pages 137–160. Prentice Hall Series in Artificial Intelligence, 2 edition, 2002.

[28] M. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks, 2003.

[29] Hui Song, Sencun Zhu, and Guohong Cao. Attack-resilient time synchronization for wireless sensor networks. In *Second IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2005)*, Washington DC, November 2005.

[30] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock synchronization in wireless sensor networks: A survey. *Ad-Hoc Networks*, 3(3):281–323, May 2005.

[31] Jana van Greunen and Jan Rabaey. Lightweight time synchronization for sensor networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 11–19, New York, NY, USA, 2003. ACM Press.