Constructing Connected-Dominating-Set with Maximum Lifetime in Cognitive Radio Networks

Zhiyong Lin[1,2], Hai Liu[2], Xiaowen Chu[2], Yiu-Wing Leung[2], Ivan Stojmenovic[3,4]

[1]Dept of Computer Science, GuangDong Polytechnic Normal University, China

[2]Dept of Computer Science, Hong Kong Baptist University

[3]SITE, University of Ottawa; [4]School of Software, Tsinghua University, Beijing 100084, China

Abstract—Connected-dominating-set (CDS) is a representative technique for constructing virtual backbones of wireless networks and thus facilitates implementation of many tasks including broadcasting, routing, etc. Most of existing works on CDS aim at constructing the minimum CDS (MCDS), so as to reduce the communication overhead over the CDS. However, MCDS may not work well in cognitive radio networks (CRNs) where communication links are prone to failure due to stochastic activities of primary users (PUs). A MCDS without consideration of the stochastic activities of PUs easily becomes invalid when the PUs become active. This study addresses a new CDS construction problem by considering the PUs' activities. Our problem is to maximize the lifetime of the CDS while minimizing the size of the CDS, where the lifetime of a CDS is defined as the expected duration that the CDS is maintained valid. We show that the problem is NP-hard and propose a three-phase centralized algorithm. Given a CRN, the centralized algorithm can compute a CDS such that the lifetime of the CDS is maximized (optimal), and the size of the CDS is upper-bounded. We further present a two-phase localized algorithm which requires 2-hop information. Extensive simulations are conducted to evaluate the proposed algorithms.

Index Terms—cognitive radio, connected-dominating-set, lifetime, fault tolerance.

1 Introduction

COGNITIVE Radio Network (CRN) has been proposed as a new kind of wireless networking paradigm, aiming at alleviating the severe scarcity in unlicensed spectrum as well as improving the efficiency of licensed spectrum usage. A CRN is a group of unlicensed users (or cognitive users, CUs) equipped with cognitive radios who coexist with the licensed users (or primary users, PUs) in the same geographic area. With cognitive radios, CUs are able to sense the licensed spectrum and opportunistically access the idle channels in the licensed spectrum of PUs without causing interference to the PUs. CUs must vacate all the related channels once these channels are reclaimed by PUs for their transmissions. In a CRN, therefore, the set of available channels for a CU dynamically changes over time due to unpredictable activities of PUs. Such a unique characteristic distinguishes CRNs from conventional wireless networks where all nodes usually operate over the same and static channels.

The concept of connected-dominating-set (CDS) plays a crucial role in the management and maintenance of wireless networks, e.g., wireless ad hoc networks. A dominating set (DS) of a given graph $\boldsymbol{G}$ is a set of nodes such that each node of $\boldsymbol{G}$ is either in the set or is adjacent to a node of the set. A CDS is defined as a connected DS. The nodes in a CDS are referred to as *dominators* and the nodes other than dominators are referred to as *dominatees*. A CDS usually serves as virtual backbone in conventional wireless networks to facilitate tasks such as broadcasting, routing, and connectivity management [1] [2]. For instance, broadcasting in a wireless ad hoc network could be simplified by letting each node of a CDS transmit the broadcast message once. Such benefit brought from CDS could be extended to the broadcasting in CRNs. Currently, study of broadcasting in CRNs is still in its infant stage (see [3] and the relevant references therein). The existing broadcasting protocols for CRNs are all non-CDS-based. These non-CDS-based approaches are complicated, and on the other hand, may not be efficient, since all CUs in the network are potentially requested to complete the broadcasting operation. CDS-based approaches provide a promising way to the broadcasting problem. For example, if a CDS of a CRN is available, we can restrict the broadcasting operation to only the CUs in the CDS and let each dominator transmit the broadcast message to all its 1-hop neighbors (Multiple transmissions may be required at each dominator in the multi-channel environment of CRNs). With the CDS-based approaches, only a small fraction of CUs is involved in the broadcasting operation which makes the operation simpler and more efficient. However, the CDS-based broadcasting approaches raise a new issue of constructing CDS in CRNs. In this study, we take the first step toward addressing the CDS construction problem in CRNs.

Extensive works have been done in constructing CDS for ad hoc networks, wireless sensor networks, and wireless mesh networks. Most of these works aim to construct the minimum CDS (MCDS), i.e., the CDS with the minimum size. Notice that the communication tasks are normally undertaken by the nodes in CDS. A CDS with the minimum size reduces the overall communication overhead and thus prolongs the network lifetime. In this sense, MCDS does work well in the wireless networks, where all the nodes operate over the same and static channels and the communication links are usually static.

However, the robustness of the MCDS is a very serious problem in CRNs, compared with conventional ad hoc networks. Notice that available channels of each CU dynamically change over time due to unpredictable activities of PUs. The communication link between two adjacent CUs is broken once there is no channel commonly-available to the CUs. Failure of communication links could cause invalidity of the previously-constructed MCDS. A CDS becomes invalid whenever 1) it is no longer connected (i.e., the dominators are not connected); or 2) it is no longer a dominating set (i.e., some dominatees cannot be dominated by the dominators). Thus, a MCDS without consideration of dynamic activities of PUs easily becomes invalid when some PUs become active and reclaim the related channels. Notice that it takes considerable communication overhead to maintain or re-construct a CDS if the CDS becomes invalid. Rather than the MCDS with the minimum size, the CDS with the maximum operation duration (lifetime) is more desired in CRNs.

This work addresses the problem of constructing the CDS with the maximum lifetime in CRNs. We assume that a PU behaves in status of ON (active on the channel) or OFF (inactive on the channel) with respective probabilities, and the time duration of the ON and OFF statuses follows a specific distribution (e.g., the exponential distribution [4] [5]). A channel is said to be available to a CU if the CU can operate over this channel without causing harmful interference to any PU being active on this channel (ON status). The communication link of two adjacent CUs is broken if there is no channel commonly-available to the CUs. In this sense, the lifetime of a communication link is the maximum duration that there exists at least one channel which is commonly-available to the endpoints of the link. Given a CRN, the problem of our concern is to compute a CDS such that the lifetime of the CDS is maximized and the size of the CDS is minimized, where the lifetime of a CDS is defined as the expected duration that the CDS is maintained valid. We prove the NP-hardness of the problem and propose a three-phase centralized algorithm as well as a two-phase localized algorithm (distributed algorithm using 2-hop neighborhood information) to the problem.

The contribution of this work is three-fold.

1) *New concept and new problem*: We introduce a new concept of lifetime for CDS which takes the stochastic activities of PUs into account. Based on this new concept, we address a new problem of maximizing the lifetime of the CDS for CRNs.

2) *New centralized algorithm with theoretically provable properties*: We propose a three-phase centralized algorithm to the problem. Given a CRN, the proposed centralized algorithm can compute a CDS, such that the lifetime of the CDS is maximized (optimal) and the size of the CDS is upper-bounded by an approximation ratio.

3) *New localized algorithm requiring 2-hop neighborhood information*: We propose a localized algorithm which requires 2-hop neighborhood information of CUs. Simulation results show that the proposed localized algorithm achieves a good trade-off between the lifetime of CDS and the size of CDS.

A preliminary version of this work was published in [6].

2 Related Work

Numerous works have been done on constructing MCDS in conventional wireless networks. Basically, the existing works can be classified into centralized [7] [8] and distributed approaches. The distributed approaches can be further divided into two categories: addition-based algorithms [9] [10] [11] [12] and pruning-based algorithms [1] [2]. Due to limited space, we move the detailed discussion to the supplementary file.

3 System Model, Definitions and Problem Formulation

We consider a CRN consisting of N, $N\geq 2$, CUs who coexist with M, $M\geq 1$, PUs in the same geographical area. The PUs independently operate over non-overlapping licensed spectrum. For simplicity, the licensed spectrum of PU m, $1\leq m\leq M$, is denoted by channel c_m. The whole set of potentially available channels is denoted by $\boldsymbol{C}=\{c_1, c_2, \cdots, c_M\}$, where channel c_m is exclusively licensed to PU m and is called channel m for convenience. We assume that each CU is equipped with one cognitive radio, by which the CU can opportunistically access the licensed spectrum of the PUs.

A CRN is modeled by a graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}=\{v_1, v_2, \cdots, v_N\}$ denotes the set of CUs (node v_i corresponding to CU i, i=1, 2, $\cdots$, N) and $\boldsymbol{E}$ is the set of communication links. We assume that PU m behaves in status of ON (active on channel c_m) or OFF (inactive on channel c_m) with respective probabilities. Channel c_m is said to be *available* to CU v_i if and only if v_i can operate over c_m without causing interference to PU m which is in ON status. There is a link $e_{i,j}$ between v_i and v_j if and only if v_i and v_j are within the transmission range of each other and there is at least one channel commonly-available to v_i and v_j. Due to the stochastic activities of PUs, the duration (lifetime) that a channel is being available on link $e_{i,j}$ is a random variable. Link $e_{i,j}$ is maintained as long as there is at least one channel available on link $e_{i,j}$. Thus, the lifetime of link $e_{i,j}$ can be determined by the expected duration of the channel which is being available on link $e_{i,j}$ for the longest time. Formally, we define the lifetime of a link as follows.

Definition 1 (Lifetime of a Link). Suppose that CUs v_i and v_j are within the transmission range of each other. The lifetime of link $e_{i,j}$ between v_i and v_j, denoted by

$\rho(e_{i,j})$, is defined as the expected value of $\max\{x_{i,j}^{(1)}, x_{i,j}^{(2)}, \cdots, x_{i,j}^{(M)}\}$ (i.e., $E(\max\{x_{i,j}^{(1)}, x_{i,j}^{(2)}, \cdots, x_{i,j}^{(M)}\})$), where random variable $x_{i,j}^{(m)} \geq 0$ denotes the duration that channel m is being commonly-available to v_i and v_j from the reference point of time (m=1, 2, $\cdots$, M).

Remarks:

1) In Definition 1, *the reference point of time* is the starting point of the lifetime of a link, and this reference point is the same to all links in the network. For example, we suppose that PUs' activities follow a probabilistic model (e.g., the exponential distribution). In centralized CRNs, at the moment that the central controller collects all required information, the controller can estimate the lifetime of each link from this moment which serves as the reference point of time. Therefore, lifetimes of all links start at the same reference point of time when their lifetimes are concerned.

2) Definition 1 gives one possible definition of the lifetime of a link. Our proposed algorithms in this work are *independent* of this definition. That is, as long as lifetime of each link (could be "average" lifetime in Definition 1 or other forms of lifetime) is determined, the proposed algorithms are valid and applicable and all the corresponding analytical results still hold. For example, the lifetime of link $e_{i,j}$ could be affected by movement of nodes v_i and v_j. In this study, we assume that the nodes are static or move with relatively low speeds and the change of channel availability is the dominating factor to the link failure, since we believe that this unique characteristic of CRNs is more important.

3) Given a probabilistic model of PUs' activities, the lifetime of a link can be determined by either analytical calculation or numerical methods (e.g., the Monte Carlo method [16]). For instance, if the time duration of the ON/OFF statuses of PUs follows the exponential distribution [4] [5], $\rho(e_{i,j})$ can be mathematically computed. Readers can refer to the Appendix in the supplementary file for more details on how to derive mathematic formulation of $\rho(e_{i,j})$. In the simulation, we adopt the Monte Carlo method to estimate lifetimes of links.

4) When PU m is in ON status, to avoid harmful interference to PU m, any CU in PU m's interference range is usually prohibited to access channel m. That is, $x_{i,j}^{(m)}$ is zero if v_i or v_j is located in PU m's interference range when PU m is active. When PU m is in OFF status, it is safe for the CUs to operate over channel m. In this case, $x_{i,j}^{(m)}$ is essentially determined by the duration of PU m being in OFF status.

Given a connected network, with each link associated with a lifetime, we define the lifetime of the network as the duration that this network is maintained connected.

Definition 2 (Lifetime of a Connected Network). Given a connected network $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the set of nodes and $\boldsymbol{E}$ is the set of links with lifetime values, the lifetime of $\boldsymbol{G}$, denoted by $\rho(\boldsymbol{G})$, is determined as follows: 1) Deleting all the links with lifetime less than $\rho(\boldsymbol{G})$ cannot cause $\boldsymbol{G}$ disconnected; 2) Deleting all the links with lifetime less than or equal to $\rho(\boldsymbol{G})$ leads to disconnection of $\boldsymbol{G}$. That is, $\rho(\boldsymbol{G})=\max\{\rho \mid$ deleting in $\boldsymbol{G}$ all the links with lifetime less than ρ cannot cause $\boldsymbol{G}$ disconnected$\}$.

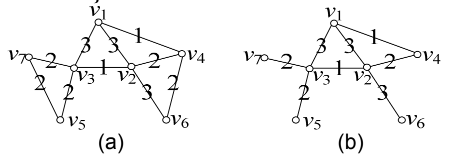


Fig. 1. An illustrative example.

Intuitively, the lifetime of a network indicates the duration that the network is maintained connected. Taking example in Fig. 1(a), the number associated with each link denotes the lifetime of the link (in time unit/slot). Notice that the lifetimes of all the links are counted starting from the same reference point of time. Lifetime of link $e_{1,4}$ (between node v_1 and node v_4) is 1. It implies that link $e_{1,4}$ is available in the 1st time slot and is broken in the 2nd time slot. Similarly, since lifetime of link $e_{2,4}$ is 2, link $e_{2,4}$ is available in time slots 1 and 2 and is broken in the 3rd time slot. After 2 time units elapse, all the links with lifetime less than or equal to 2 are assumed to be broken and the network consequently becomes disconnected. That is, the connected topology in Fig. 1(a) can last for at most 2 time units. It is easy to identify that the lifetime of the network in Fig. 1(a) is 2 (time units).

Based on Definitions 1 and 2, we introduce a new concept of lifetime for CDS. Notice that a CDS becomes invalid whenever 1) it is no longer connected, i.e., the *internal* connectivity of the dominators is not preserved; or 2) it is no longer a dominating set, i.e., some *external* links (between the dominators and the dominatees) are broken and some dominatees cannot be dominated by the dominators. Therefore, the lifetime of a CDS not only depends on its *internal lifetime* (the robustness of its internal connectivity), but also its *external lifetime* (the robustness of its external links). Accordingly, we have the following definitions.

Definition 3 (Internal Lifetime of a CDS). Let $\boldsymbol{S}$ be a CDS of $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$. The internal lifetime of $\boldsymbol{S}$ is defined as $\rho_{\text{in}}(\boldsymbol{S})=\rho(\boldsymbol{G}[\boldsymbol{S}])$, where $\boldsymbol{G}[\boldsymbol{S}]$ denotes the subgraph of $\boldsymbol{G}$ induced by $\boldsymbol{S}$[1].

Definition 4 (External Lifetime of a CDS). Let $\boldsymbol{S}$ be a CDS of $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$. The external lifetime of $\boldsymbol{S}$ is defined as $\rho_{\text{ex}}(\boldsymbol{S})=\min\{\tau(v) \mid v\in \boldsymbol{V}-\boldsymbol{S}\}$, where $\tau(v)=\max\{\rho(e) \mid e\in\boldsymbol{E}$, one endpoint of e is dominatee v and the other endpoint is a dominator in $\boldsymbol{S}\}$, which denotes the longest duration that dominatee v can be dominated by $\boldsymbol{S}$.

We use Fig. 1(a) to illustrate Definitions 3 and 4. Let $\boldsymbol{S}$=$\{v_1, v_2, v_3\}$ be a CDS of $\boldsymbol{G}$, i.e., any node of $\boldsymbol{G}$ is either in $\boldsymbol{S}$ or is adjacent to a node in $\boldsymbol{S}$. Nodes of $\{v_1, v_2, v_3\}$ are dominators and nodes of $\{v_4, v_5, v_6, v_7\}$ are domi-

[1] In the graph theory, given $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ and $\boldsymbol{S}\subseteq\boldsymbol{V}$, the *induced subgraph* $\boldsymbol{G}[\boldsymbol{S}]$ consists of all the nodes in $\boldsymbol{S}$ and all edges in $\boldsymbol{E}$ whose endpoints are in $\boldsymbol{S}$.

natees. The subgraph induced by $\boldsymbol{S}$, i.e. $\boldsymbol{G}[\boldsymbol{S}]$, contains three nodes $\{v_1, v_2, v_3\}$ and three links $\{e_{1,2}, e_{1,3}, e_{2,3}\}$. It is easy to identify that the lifetime of $\boldsymbol{G}[\boldsymbol{S}]$ is 3. According to Definition 3, the internal lifetime of CDS $\boldsymbol{S}$, i.e. $\rho_{in}(\boldsymbol{S})$, is equal to 3. That is, nodes in $\boldsymbol{S}$ are no longer connected after 3 time units elapse. Node v_4 is a dominatee and is dominated by dominators v_1 and v_2. We have $\tau(v_4)$=max$\{\rho(e_{1,4}), \rho(e_{2,4})\}$=max$\{1, 2\}$=2. That is, v_4 can no longer be dominated by CDS $\boldsymbol{S}$ after 2 time units elapse. Similarly, we can identify that $\tau(v_5)$=2, $\tau(v_6)$=3, and $\tau(v_7)$=2. Hence, according to Definition 4, the external lifetime of CDS $\boldsymbol{S}$, i.e. $\rho_{ex}(\boldsymbol{S})$, is equal to min$\{\tau(v_4), \tau(v_5), \tau(v_6), \tau(v_7)\}$=2. It implies that at least one dominatee cannot be dominated by any dominators ($\boldsymbol{S}$ becomes invalid) after 2 time units elapse.

A CDS becomes invalid when either its internal lifetime or its external lifetime expires. We define the lifetime of a CDS as the smaller one of its internal lifetime and external lifetime to ensure the validity of the CDS.

Definition 5 (Lifetime of a CDS). The lifetime of a CDS $\boldsymbol{S}$ is defined as $\rho(\boldsymbol{S})$=min$\{\rho_{in}(\boldsymbol{S}), \rho_{ex}(\boldsymbol{S})\}$.

The lifetime of CDS $\boldsymbol{S}$ $\rho(\boldsymbol{S})$ is not necessarily computed by $\rho_{in}(\boldsymbol{S})$ and $\rho_{ex}(\boldsymbol{S})$. In fact, $\rho(\boldsymbol{S})$ can be easily determined by the subgraph spanned by $\boldsymbol{S}$ which is defined as follows.

Definition 6 (Subgraph Spanned by a CDS). Let $\boldsymbol{S}$ be a CDS of $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$. The subgraph spanned by $\boldsymbol{S}$, denoted by $\boldsymbol{G}$<$\boldsymbol{S}$>, is the graph which contains all nodes in $\boldsymbol{V}$ and the only links associated with at least one endpoint in $\boldsymbol{S}$. That is, $\boldsymbol{G}$<$\boldsymbol{S}$>=$(\boldsymbol{V}, \boldsymbol{E}')$, where $\boldsymbol{E}'=\{ e \,|\, e\in\boldsymbol{E}$, at least one endpoints of e is a dominator in $\boldsymbol{S}\}$.

The following theorem states that the lifetime of a CDS equals the lifetime of the subgraph spanned by the CDS.

Theorem 1. *Let* $\boldsymbol{S}$ *be a CDS of* $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$. *We have* $\rho(\boldsymbol{S})=\rho(\boldsymbol{G}$<$\boldsymbol{S}$>$)$, *where* $\boldsymbol{G}$<$\boldsymbol{S}$> *is the subgraph spanned by* $\boldsymbol{S}$.

Proof. Due to limited space, the proof is moved to the supplementary file. □

Taking example in Fig. 1(a) again, $\boldsymbol{S}=\{v_1, v_2, v_3\}$ is a CDS with $\rho_{in}(\boldsymbol{S})$=3 and $\rho_{ex}(\boldsymbol{S})$=2 according to Definitions 3 and 4. According to Definition 5, we have $\rho(\boldsymbol{S})$=min$\{\rho_{in}(\boldsymbol{S}), \rho_{ex}(\boldsymbol{S})\}$=2, i.e., the lifetime of CDS $\boldsymbol{S}$ is 2. Actually, after 2 time units elapse, links $e_{1,4}$ and $e_{2,4}$ are broken and thus dominatee v_4 can no longer be dominated ($\boldsymbol{S}$ becomes invalid). The subgraph spanned by $\boldsymbol{S}$ is shown in Fig. 1(b). According to Definition 2, its lifetime is 2 which equals the lifetime of $\boldsymbol{S}$. In this example, $\{v_2, v_3\}$ is MCDS with lifetime 1 while CDS $\{v_1, v_2, v_3\}$ is with the maximum lifetime 2.

In this work, we study the problem of constructing CDS with lifetime maximized in CRNs. In CRNs, availability of a link and duration of a link being available (i.e., lifetime of a link) are totally random. Lifetime of a CDS depends on lifetimes of links and thus is also a random variable. Since lifetime of a CDS is essentially random, any CDS that is computed by any algorithm may have very short lifetime in practice. Given this randomness, we believe that maximizing "average" lifetime of a CDS (Definition 5) is one of the proper ways to build a robust CDS. Furthermore, notice that two CDSs with the same lifetime may have different internal lifetimes and external lifetimes. When the external lifetime expires, individual dominatee cannot be dominated, which is a local effect. In contrast, when the internal lifetime expires, the whole CDS is disconnected and thus cannot function properly, which is a global effect. Therefore, we believe that the internal lifetime is more important than the external lifetime. There is a trade-off between the lifetime and the size of CDS. In general, a CDS with larger size (i.e., more nodes and more links) is more robust in terms of connectivity. However, increasing size of a CDS will increase the overall energy consumption of the CDS since CDS nodes normally spend more energy than non-CDS nodes. Node rotation in a CDS with small size will balance energy consumption and eventually prolong the network lifetime. Our problem is formally formulated as follows.

Maximizing Lifetime of CDS (MLCDS). Given a CRN $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ and the stochastic activities of PUs, the MLCDS problem is to compute a CDS $\boldsymbol{S}$ of $\boldsymbol{G}$ such that: 1) The lifetime of $\boldsymbol{S}$ is maximized, i.e., max $\rho(\boldsymbol{S})$; 2) The internal lifetime of $\boldsymbol{S}$ is maximized, i.e., max $\rho_{in}(\boldsymbol{S})$; 3) The size of $\boldsymbol{S}$ is minimized, i.e., min $|\boldsymbol{S}|$.

The MLCDS problem is a multi-objective optimization problem. The three objectives are difficult to be optimized simultaneously. In practice, a multi-objective problem is usually tackled by optimizing the objectives sequentially. In CRNs, we believe that the lifetime of a CDS is more important than the size. It is because that a CDS with short lifetime is prone to failure, which may cause considerable communication overhead to maintain and re-construct the CDS. Therefore, we sequentially optimize the three objectives in the MLCDS problem. Specifically, we first maximize the lifetime and the internal lifetime of the CDS and then minimize the size of the CDS with the maximum lifetime preserved.

The following theorem shows that the MLCDS problem is NP-hard.

Theorem 2. *The MLCDS problem is NP-hard.*

Proof. The proof is moved to the supplementary file. □

4 A THREE-PHASE CENTRALIZED ALGORITHM

In this section, we propose a three-phase centralized algorithm to the MLCDS problem. The centralized algorithm is suitable to the CRNs with centralized structures (i.e., infrastructure-based). A recent study [17] pointed out that centralized structures are required in CRNs to solve the problems of security fragility, spectral inefficiency, and high terminal cost. Given a CRN $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ and the stochastic activities of PUs, the lifetime of each link in $\boldsymbol{G}$ is first determined (see the Appendix in the supplementary file). With each link associated with a lifetime, our task is to sequentially optimize the three objectives of the MLCDS problem, i.e., maximizing the lifetime, maximizing the internal lifetime, and minimizing the size of the CDS. Accordingly, our algorithm consists of three phases which aim at the three objectives, respectively. Given $\boldsymbol{G}$, in the first phase, we reduce the searching space by computing a connected subgraph $\boldsymbol{G}'$ of $\boldsymbol{G}$, such that CDS S of $\boldsymbol{G}$ has the maximum lifetime if and only if S is a CDS of $\boldsymbol{G}'$. That is, the CDS of $\boldsymbol{G}$ with the maximum lifetime is guaranteed to be included in subgraph $\boldsymbol{G}'$. Based on $\boldsymbol{G}'$, in the second phase, we compute a subgraph $\boldsymbol{G}''$of $\boldsymbol{G}'$ such that any CDS of $\boldsymbol{G}'$ (also the CDS of $\boldsymbol{G}$) constructed within $\boldsymbol{G}''$ can further maximize the internal lifetime with the maximum lifetime preserved. Finally, in the third phase, we compute a CDS of $\boldsymbol{G}'$ within $\boldsymbol{G}''$ and minimize its size, with the maximum lifetime and the maximum internal lifetime both preserved. It should be emphasized that our algorithm is independent of how the lifetime of each link in $\boldsymbol{G}$ is calculated. In other words, our algorithm is applicable to any connected graph $\boldsymbol{G}$ with each link associated with a lifetime.

For ease of description, given two sets $S_1 \subseteq \boldsymbol{V}$ and $S_2 \subseteq \boldsymbol{V}$ in graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, hereafter we say S_2 is dominated by S_1 if each node in S_2 either belongs to S_1 or is adjacent to a node in S_1.

4.1 Phase One: Maximize Lifetime

```
MaxLifetime Algorithm
INPUT: a connected graph $G(V, E)$ in which each link is associated with a lifetime.
OUTPUT: $G'=(V, E')$, a connected subgraph of $G$.
1: Sort the lifetimes $\{\rho(e) | e \in E\}$ in ascending order and get $l$ lifetime levels: $\rho_1<\rho_2<\cdots<\rho_l$;
2: Initialize $E'=E$;
3: FOR $i$=1, 2, $\cdots$, $l$   // link deletion
4:     $E_i=\{e | \rho(e)=\rho_i, e \in E'\}$;   // set of the links with lifetime $\rho_i$
5:     IF graph($V$, $E'-E_i$) still keeps connected
6:         $E'=E'-E_i$;   // delete all the links with lifetime $\rho_i$
7:     ELSE BREAK;   // exit the for-loop
8: RETURN $G'=(V, E')$;
```

Fig. 2. Pseudo code of the *MaxLifetime* algorithm.

According to Theorem 1, the lifetime of CDS S of $\boldsymbol{G}$, i.e. $\rho(S)$, is equal to $\rho(\boldsymbol{G}<S>)$, where $\boldsymbol{G}<S>$ denotes the subgraph of $\boldsymbol{G}$ spanned by S. That is, maximizing the lifetime of S is equivalent to maximizing the lifetime of $\rho(\mathbf{G}<S>)$. On the other hand, according to Definition 2, lifetime of a connected network (graph) is determined by the lifetimes of links in the graph. Therefore, to construct the CDS of $\boldsymbol{G}$ with the maximum lifetime, we should avoid selecting those links with small lifetimes. Inspired by this intuitive observation, we design a pruning-based algorithm, called *MaxLifetime*, which can generate a subgraph $\boldsymbol{G}'$ of $\boldsymbol{G}$ such that any CDS of $\boldsymbol{G}'$ is the CDS of $\boldsymbol{G}$ with the maximum lifetime. In this sense, we reduce the searching space from $\boldsymbol{G}$ to $\boldsymbol{G}'$. The algorithm works as follows. Given a connected graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, we first sort all the links of $\boldsymbol{G}$ in ascending order according to their lifetimes. Note that the links may have the same lifetime. Suppose that there are l different lifetime levels, which are denoted by $\rho_1<\rho_2<\cdots<\rho_l$. Let $\boldsymbol{G}'=(\boldsymbol{V}, \boldsymbol{E}')$ be a subgraph of $\boldsymbol{G}$ with $\boldsymbol{E}'$ initialized as $\boldsymbol{E}$. Then, we continually delete the links in $\boldsymbol{E}'$ according to lifetime level from ρ_1 to ρ_l as long as such link deletion will not cause disconnection of $\boldsymbol{G}'$. When the algorithm ends, we get a connected subgraph $\boldsymbol{G}'$ of $\boldsymbol{G}$. The pseudo code of the *MaxLifetime* algorithm is presented in Fig .2.

The following lemma shows the time complexity of the *MaxLifetime* algorithm.

Lemma 1. *The time complexity of the MaxLifetime algorithm is $O(|\boldsymbol{V}|^4)$.*

Proof. The proof is moved to the supplementary file. □

Given graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, the *MaxLifetime* algorithm can generate a connected subgraph $\boldsymbol{G}'(\boldsymbol{V}, \boldsymbol{E}')$ of $\boldsymbol{G}$. Let $\eta=\min\{\rho(e) | e \in \boldsymbol{E}'\}$ be the smallest lifetime of links in $\boldsymbol{G}'$ and $\rho^*=\max\{\rho(S) | S$ is a CDS of $\boldsymbol{G}\}$ be the maximum lifetime of CDSs of $\boldsymbol{G}$. The following lemma indicates that both $\boldsymbol{G}$ and $\boldsymbol{G}'$ have the same lifetime, which is equal to η, and this value also equals ρ^*.

Lemma 2. *Given $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, let $\boldsymbol{G}'(\boldsymbol{V}, \boldsymbol{E}')$ be the output subgraph by the MaxLifetime algorithm. We have: $\rho(\boldsymbol{G})=\rho(\boldsymbol{G}')=\eta=\rho^*$, where $\eta=min\{\rho(e) | e \in \boldsymbol{E}'\}$ and $\rho^*=max\{\rho(S) | S$ is a CDS of $\boldsymbol{G}\}$.*

Proof. The proof is moved to the supplementary file. □

Based on Lemma 2, we prove that the subgraph $\boldsymbol{G}'$ computed by the *MaxLifetime* algorithm for $\boldsymbol{G}$ has the following property: constructing CDS of $\boldsymbol{G}'$ is equivalent to constructing CDS of $\boldsymbol{G}$ with the maximum lifetime. This result is formally presented in the following theorem.

Theorem 3. *Given $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, let $\boldsymbol{G}'(\boldsymbol{V}, \boldsymbol{E}')$ be the output subgraph by the MaxLifetime algorithm. We have: $S \subseteq \boldsymbol{V}$ is a CDS of $\boldsymbol{G}$ with the maximum lifetime (i.e., $\rho(S)=\rho^*$) if and only if S is a CDS of $\boldsymbol{G}'$.*

Proof. The proof is moved to the supplementary file. □

4.2 Phase Two: Maximize Internal Lifetime

After the optimization in phase one, we obtain a connected subgraph $\boldsymbol{G}'(\boldsymbol{V}, \boldsymbol{E}')$ of $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$. Any CDS of $\boldsymbol{G}'$ is also a CDS of $\boldsymbol{G}$. According to Theorem 3, to construct the CDS of $\boldsymbol{G}$ with the maximum lifetime, we only need to construct a CDS of $\boldsymbol{G}'$ instead. Since the lifetime of a CDS is the minimum of its internal lifetime and external lifetime, the CDS with the maximum lifetime does not necessarily achieve the maximum internal lifetime. Hence, we pursue optimization in this section to further maximize the internal lifetime with the maximum lifetime preserved.

Given a CDS S of $\boldsymbol{G}'$, according to Definition 3, its internal lifetime $\rho_{\text{in}}(S)$ is determined by $\rho(\boldsymbol{G}'[S])$, where $\boldsymbol{G}'[S]$ is the subgraph of $\boldsymbol{G}'$ induced by S. To maximize $\boldsymbol{G}'[S]$, intuitively, we should ensure that the nodes in S are connected via the links with lifetime as large as possible. The basic idea of this phase is similar to that in phase one, i.e., the links with small lifetime are pruned to achieve large internal lifetime. We design a greedy algorithm called *MaxInternalLifetime* in this section. The proposed algorithm can generate a subgraph G'' of the given graph G' such that any CDS of G' constructed within G'' can achieve the maximum internal lifetime. The algorithm works in a similar way as the *MaxLifetime* algorithm. Specifically, given $G'(V, E')$, we first sort the links of G' in ascending order according to their lifetimes. Suppose that there are l different lifetime levels, which are denoted by $\rho_1<\rho_2<\cdots<\rho_l$. Let $G''=(V, E'')$ be a subgraph of G' with E'' initialized as E'. Then, we try to delete the links in E'' according to lifetime level from ρ_1 to ρ_l. The link deletion terminates when any further link deletion would result in that no component[2] of G'' can still include a CDS of G'. When the algorithm ends, we get a subgraph G'' of G' in which there exists at least one component that contains a CDS of G'. The pseudo code of the *MaxInternalLifetime* algorithm is presented in Fig. 3.

```
MaxInternalLifetime Algorithm
INPUT: a connected graph $G'(V, E')$ in which each link is associated with a lifetime.
OUTPUT: $G''=(V, E'')$, a subgraph of $G'$.
1: Sort the lifetimes $\{\rho(e)\,|\,e\in E'\}$ in ascending order and get $l$ lifetime levels: $\rho_1<\rho_2<\cdots<\rho_l$;
2: Initialize $E''=E'$;
3: FOR $i=1, 2, \cdots, l$   // link deletion
4:     $E_i=\{e\,|\,\rho(e)=\rho_i, e\in E''\}$;   // set of the links with lifetime $\rho_i$
5:     IF graph($V, E''-E_i$) still has a component containing a CDS of $G'$
6:         $E''=E''-E_i$;   // delete all the links with lifetime $\rho_i$
7.     ELSE BREAK;   // exit the for-loop
8. RETURN $G''=(V, E'')$;
```

Fig. 3. Pseudo code of the *MaxInternalLifetime* algorithm.

The following lemma shows the time complexity of the *MaxInternalLifetime* algorithm.

Lemma 3. *The time complexity of the MaxInternalLifetime algorithm is $O(|V|^4)$.*

Proof. The proof is moved to the supplementary file. □

Given graph $G'(V, E')$, the *MaxInternalLifetime* algorithm generates a subgraph $G''(V, E'')$ of G'. Let $\gamma=\min\{\rho(e)\,|\,e\in E''\}$ be the smallest lifetime of links in G'' and $\rho_{\text{in}}^*=\max\{\rho_{\text{in}}(S)\,|\,S$ is a CDS of $G'\}$ be the maximum internal lifetime of CDSs of G', respectively. The following lemma shows that γ is exactly equal to ρ_{in}^*.

Lemma 4. *Given $G'(V, E')$, let $G''(V, E'')$ be the output subgraph by the MaxInternalLifetime algorithm. We have: $\gamma=\rho_{\text{in}}^*$, where $\gamma=\min\{\rho(e)\,|\,e\in E''\}$ and $\rho_{\text{in}}^*=\max\{\rho_{\text{in}}(S)\,|\,S$ is a CDS of $G'\}$.*

Proof. The proof is moved to the supplementary file. □

Based on Lemma 4, we obtain the following theorem, which shows that the *MaxInternalLifetime* algorithm is guaranteed to generate a subgraph G'' such that 1) any CDS of G' achieves the maximum internal lifetime if it is connected in G'', and 2) any CDS of G' with the maximum internal lifetime is contained in a component of G''.

Theorem 4. *Given $G'(V, E')$, let $\rho_{\text{in}}^*=\max\{\rho_{\text{in}}(S)\,|\,S$ is a CDS of $G'\}$ and $G''(V, E'')$ be the output subgraph by the MaxInternalLifetime algorithm applied on G'. We have: 1) If $S\subseteq V$ is a CDS of G' and $G''[S]$ is connected, then $\rho_{\text{in}}(S)=\rho_{\text{in}}^*$; 2) If S is a CDS of G' and $\rho_{\text{in}}(S)=\rho_{\text{in}}^*$, then $G''[S]$ is connected.*

Proof. The proof is moved to the supplementary file. □

Given graph $G(V, E)$, we perform the *MaxLifetime* algorithm in phase one and obtain a connected subgraph $G'(V, E')$ of G. Based on $G'(V, E')$, we perform the *MaxInternalLifetime* algorithm in phase two and obtain a subgraph $G''(V, E'')$ of G'. Theorem 3 shows that constructing CDS of G with the maximum lifetime can be transferred to constructing CDS of G'. Theorem 4 shows that constructing CDS of G' within G'' can maximize the internal lifetime of the CDS. Furthermore, according to the *MaxLifetime* algorithm, a CDS of G' with the maximum internal lifetime is also the CDS of G with the maximum internal lifetime (both maximums are the same). Based on these observations, we obtain the following theorem.

Theorem 5. *Given $G(V, E)$, let $G'(V, E')$ be the output subgraph by the MaxLifetime algorithm applied on G. Let $G''(V, E'')$ be the output subgraph by the MaxInternalLifetime algorithm applied on G'. We have: $S\subseteq V$ is a CDS of G with the maximum lifetime and the maximum internal lifetime if and only if S is a DS of G' and $G''[S]$ is connected.*

Proof. The proof is moved to the supplementary file. □

4.3 Phase Three: Minimize Size with Maximum Lifetime Preserved

Given graph $G(V, E)$, after optimization in phase one and phase two, we obtain the subgraphs $G'(V, E')$ and $G''(V, E'')$, respectively. Notice that any CDS of G' constructed within G'' can achieve the maximum lifetime as well as the maximum internal lifetime. In phase three, the remaining problem is to compute a MCDS of G' in G'' with the maximum lifetime and the maximum internal lifetime both preserved. We point out that the problem involved in this phase is different to the conventional MCDS problem. Our problem in this phase is to compute a CDS of G' within G'' which is a subgraph of G', while the MCDS problem has no such restriction. To distinguish from the conventional MCDS problem, we refer to the problem in phase three as *MCDS with Restriction ($MCDS_R$) Problem*, which is formally described as follows.

MCDS_R Problem. Given graph G' and its subgraph G'', the MCDS_R problem is to find a subset of nodes, say S, which satisfies the following requirements: 1) Node set S is a DS of G'; 2) The subgraph induced by S in G'', i.e. $G''[S]$, is connected; 3) The size of S, i.e. $|S|$, is minimized.

Notice that G' is the subgraph computed by the *MaxLifetime* algorithm applied on G (in phase one) and G'' is the subgraph computed by the *MaxInternalLife-*

[2] In the graph theory, given graph G and its subgraph A, A is called a component of G if 1) A is connected and 2) there is no other connected subgraph B of G such that B contains A.

time algorithm applied on $\boldsymbol{G'}$ (in phase two). According to Theorem 5, requirements 1) and 2) in the MCDS$_R$ problem ensure that $\boldsymbol{S}$ is a CDS of $\boldsymbol{G}$ with the maximum lifetime and the maximum internal lifetime preserved. Requirement 3) ensures that the size of $\boldsymbol{S}$ is minimized. For convenience, hereafter we use MCDS$_R$($\boldsymbol{G'}$, $\boldsymbol{G''}$) to denote the MCDS of $\boldsymbol{G'}$ with restriction in $\boldsymbol{G''}$. The following theorem shows the NP-hardness of the MCDS$_R$ problem.

Theorem 6. *The MCDS$_R$ problem is NP-hard.*

Proof. The proof is moved to the supplementary file. □

We next present an approximation algorithm, called *ConstructMCDS$_R$*, to the MCDS$_R$ problem. The basic idea of the algorithm is as follows. Given $\boldsymbol{G'}(\boldsymbol{V}, \boldsymbol{E'})$ and its subgraph $\boldsymbol{G''}$, we check each component $\boldsymbol{g_i}$ of $\boldsymbol{G''}$. We skip $\boldsymbol{g_i}$ if the nodes in $\boldsymbol{g_i}$ cannot dominate $\boldsymbol{V}$, i.e., $\boldsymbol{g_i}$ does not contain a CDS of $\boldsymbol{G'}$. Otherwise, we attempt to compute a MCDS of $\boldsymbol{g_i}$, i.e., MCDS($\boldsymbol{g_i}$). If MCDS($\boldsymbol{g_i}$) can dominate $\boldsymbol{V}$, we let S_i=MCDS($\boldsymbol{g_i}$) which is also a CDS of $\boldsymbol{G'}$. Otherwise, additional nodes in $\boldsymbol{g_i}$ need to be added into S_i for dominating $\boldsymbol{V}$. To minimize the number of nodes in the CDS, we add as few as possible nodes to dominate $\boldsymbol{V}$. This can be done by employing any existing algorithm to the minimum-set-cover problem [18]. The nodes in MCDS($\boldsymbol{g_i}$) along with these additional nodes are included into $\boldsymbol{S_i}$, which is a CDS of $\boldsymbol{G'}$. Finally, we output $\boldsymbol{S}$ which is the $\boldsymbol{S_i}$ with the smallest size. The pseudo code of the *ConstructMCDS$_R$* algorithm is described in Fig. 4.

```
ConstructMCDS_R Algorithm
INPUT: $G'(V, E')$ and its subgraph $G''$ consisting of $K$
       components $g_i(V_i'', E_i'')$ ($i$=1, 2, ⋯, $K$).
OUTPUT: $S$ which is a subset of $V$ and a CDS of $G'$.
 1: $S=V$;
 2: FOR $i$=1, 2, ⋯, $K$
 3:   IF $V_i''$ can dominate $(V-V_i'')$ in $G'$
 4:     $X_i$=ConstructMCDS($g_i$);    // comment 1
 5:     IF $X_i$ can dominate $(V-V_i'')$ in $G'$
 6:       $S_i=X_i$;
 7:     ELSE
 8:       $U=\{v \mid v\in(V-V_i'')$ and $v$ is not dominated by $X_i\}$;
 9:       $Y_i$=FindMSC($V_i''-X_i$, $U$);    // comment 2
10:       $S_i=X_i+Y_i$;
11:     IF $|S_i|<|S|$
12:       $S=S_i$;
13: RETURN $S$;
```

Fig. 4. Pseudo code of the *ConstructMCDS$_R$* algorithm.

Comments: 1) ConstructMCDS(g) is the subroutine to find MCDS of graph g and it return a CDS of g (e.g., we can use the approach proposed by Guha *et al.* in [7]); 2) FindMCS(V'', $\boldsymbol{U}$) is the subroutine to find a minimum number of nodes in V'' to cover/dominate all nodes in $\boldsymbol{U}$ and it return a subset of V'' (e.g., we can use the well-known greedy algorithm for the minimum-set-cover problem [18]).

The following lemma shows the time complexity of the *ConstructMCDS$_R$* algorithm.

Lemma 5. *The time complexity of the ConstructMCDS$_R$ algorithm is $O(|\boldsymbol{V}|^4)$.*

Proof. The proof is moved to the supplementary file. □

We analyze the approximation ratio of the *ConstructMCDS$_R$* algorithm in the following theorem.

Theorem 7. *Let α and β denote the approximation ratios of any algorithms to the MCDS problem and the minimum-set-cover problem, respectively. The ConstructMCDS$_R$ algorithm achieves the approximation ratio not greater than $(\alpha+\beta)$.*

Proof. The proof is moved to the supplementary file. □

Corollary 1. *The ConstructMCDS$_R$ algorithm can achieve the approximation ratio not greater than $(2.78+ln\Delta+ln|\boldsymbol{V}|-lnln|\boldsymbol{V}|)$, where Δ is the maximum node degree of the input graph $\boldsymbol{G'}(\boldsymbol{V}, \boldsymbol{E'})$.*

Proof. The proof is moved to the supplementary file. □

4.4 Overall Algorithm and Numerical Example

The overall algorithm, denoted by *C-MLCDS* (centralized algorithm to the MLCDS problem), is shown in Fig. 5.

```
C-MLCDS Algorithm
INPUT: $G(V, E)$ in which each link is associated with a life-
       time.
OUTPUT: $S$ which is a CDS of $G$.
1: $G'(V, E')$=MaxLifetime($G$);
2: $G''(V, E'')$=MaxInternalLifetime($G'$);
3: $S$=ConstructMCDS_R($G'$, $G''$);
4: RETURN $S$;
```

Fig. 5. Pseudo code of the *C-MLCDS* algorithm.

We give a numerical example to demonstrate the overall algorithm. The given graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ is shown in Fig. 6(a), where there are 8 nodes in $\boldsymbol{V}$ and the lifetimes of links in $\boldsymbol{E}$ vary from 1 to 3 (time units). The number associated with each link denotes the lifetime of the link. We apply our three-phase *C-MLCDS* algorithm on the graph.

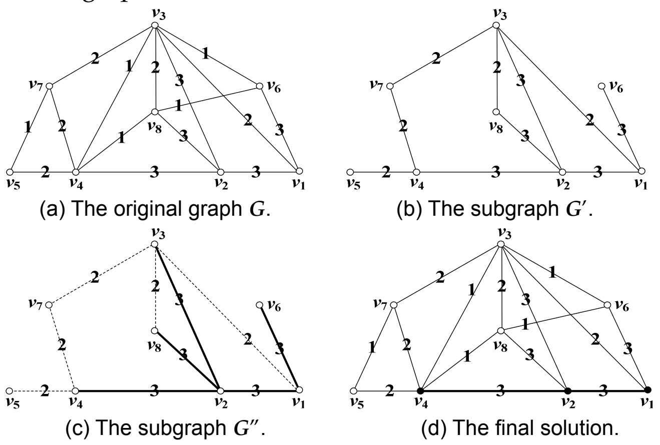


Fig. 6. A numerical example for the *C-MLCDS* algorithm.

Phase one: $\boldsymbol{G'}$=MaxLifetime($\boldsymbol{G}$). In the first phase, we call the *MaxLifetime* algorithm to generate $\boldsymbol{G'}$ which is a connected subgraph of $\boldsymbol{G}$. As shown in Fig. 6(b), $\boldsymbol{G'}$ is obtained from $\boldsymbol{G}$ by pruning all the links with lifetime 1, since further pruning the links with lifetime 2 will cause disconnection of $\boldsymbol{G'}$.

Phase two: $\boldsymbol{G''}$=MaxInternalLifetime($\boldsymbol{G'}$). In the second phase, we call the *MaxInternalLifetime* algorithm to generate $\boldsymbol{G''}$ which is a subgraph of $\boldsymbol{G'}$. As shown in Fig. 6(c), $\boldsymbol{G''}$ is obtained from $\boldsymbol{G'}$ by pruning all the links with lifetime 2 (see the dashed links). There are three components in $\boldsymbol{G}''$, i.e., g_1 consisting of $\{v_1, v_2, v_3, v_4, v_6, v_8\}$, g_2 consisting of $\{v_5\}$, and g_3 consisting of $\{v_7\}$. Only component g_1 contains the CDS of $\boldsymbol{G}'$. Clearly, there is no component of $\boldsymbol{G}''$ which contains the CDS of $\boldsymbol{G}'$ if we further prune the links with lifetime 3.

Phase three: $\boldsymbol{S}$=ConstructMCDS$_R$($\boldsymbol{G}'$, $\boldsymbol{G}''$). Based on $\boldsymbol{G}'$ and $\boldsymbol{G}''$, the third phase is to call the *ConstructMCDS$_R$* algorithm to compute MCDS$_R$($\boldsymbol{G}'$, $\boldsymbol{G}''$). Notice that only component g_1 of $\boldsymbol{G}''$ contains a CDS of $\boldsymbol{G}'$. According to the *ConstructMCDS$_R$* algorithm, we first construct the MCDS of g_1 by using a MCDS-constructing method, such as the one proposed by Guha *et al.* in [7]. Applying this algorithm on g_1, we can easily obtain the corresponding CDS of g_1, i.e. $\boldsymbol{S}=\{v_1, v_2\}$. Obviously, $\boldsymbol{S}$ is not a CDS of $\boldsymbol{G}'$. So, we select additional nodes from $\{v_3, v_4, v_6, v_8\}$ and add them into $\boldsymbol{S}$ to form a CDS of $\boldsymbol{G}'$. According to the *ConstructMCDS$_R$* algorithm, this is actually reduced to solving the minimum-set-cover problem, i.e., to use a minimal subset of $\{v_3, v_4, v_6, v_8\}$ to cover $\{v_5, v_7\}$ which is not yet covered by current $\boldsymbol{S}$. By employing greedy strategy, we can find that $\{v_4\}$ is sufficient to cover $\{v_5, v_7\}$. That is, we should add node v_4 into $\boldsymbol{S}$ and obtain $\boldsymbol{S}=\{v_1, v_2\}\cup\{v_4\}=\{v_1, v_2, v_4\}$. The final solution $\boldsymbol{S}$ is $\{v_1, v_2, v_4\}$, as shown in Fig. 6(d).

Clearly, $\boldsymbol{S}$ is a CDS of $\boldsymbol{G}$, with its internal lifetime $\rho_{in}(\boldsymbol{S})$ and external lifetime $\rho_{ex}(\boldsymbol{S})$ equal to 3 and 2, respectively. According to Definition 5, lifetime of $\boldsymbol{S}$ is 2. Adding any other nodes into $\boldsymbol{S}$ cannot increase its lifetime as a valid CDS. For example, lifetime of link $e_{4,5}$ (between node v_4 and node v_5) is 2, which implies that link $e_{4,5}$ is expected to be broken after two time units. Since lifetime of link $e_{5,7}$ is 1, link $e_{5,7}$ is weaker than link $e_{4,5}$ and link $e_{5,7}$ is broken (after one time unit) before link $e_{4,5}$ is broken (after two time units). Thus, adding node v_7 into $\boldsymbol{S}$ does not help improve robustness of $\boldsymbol{S}$ in covering node v_5. It can be verified that $\boldsymbol{S}=\{v_1, v_2, v_4\}$ achieves the maximum lifetime. Notice that, as an efficient centralized algorithm for constructing MCDS, Guha's algorithm generates the CDS $\{v_3, v_4\}$ of $\boldsymbol{G}$, which has lifetime equal to 1.

The following theorem presents the time complexity of the *C-MLCDS* algorithm.

Theorem 8. *The time complexity of the C-MLCDS algorithm is $O(|\boldsymbol{V}|^4)$.*

Proof. The proof is moved to the supplementary file. □

Let α^* and β^* denote the theoretically best approximation ratio for the MCDS problem and that for the minimum-set-cover problem, respectively. The following theorem shows that any algorithm to the MLCDS problem cannot achieve approximation ratio lower than $\min\{\alpha^*, \beta^*\}$ with respect to the size of the CDS.

Theorem 9. *Regarding the size of CDS in the MLCDS problem, approximation ratio less than $\min\{\alpha^*, \beta^*\}$ is not achievable.*

Proof. The proof is moved to the supplementary file. □

Theorem 10. *For the MLCDS problem, the C-MLCDS algorithm can compute the CDS such that 1) the optimality on both the lifetime and the internal lifetime is achieved; and 2) the approximation ratio on the size is not greater than $(\alpha^*+\beta^*)$.*

Proof. The proof is moved to the supplementary file. □

Corollary 2. *Regarding the size of the CDS in the MLCDS problem, the C-MLCDS algorithm can achieve the approximation ratio not greater than $(2.78+\ln\Delta+\ln|\boldsymbol{V}|-\ln\ln|\boldsymbol{V}|)$, where Δ is the maximum degree of the input graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$.*

Proof. The proof is moved to the supplementary file. □

5 Distributed Algorithm Using 2-Hop Neighborhood Information

The proposed *C-MLCDS* algorithm is centralized, which can be implemented by servers in infrastructure-based CRNs. However, in infrastructureless CRNs (such as CRAHNs [19]) which are lack of central controllers, distributed algorithms are usually preferred. To this aim, in this section we devise a localized algorithm (distributed algorithm using 2-hop neighborhood information) for constructing MLCDS, which is termed *L-MLCDS* (localized algorithm to MLCDS) in contrast to *C-MLCDS* (centralized algorithm to MLCDS) presented in Section 4.

5.1 *L-MLCDS* Algorithm

The desired *L-MLCDS* algorithm should have high degree of localization (i.e., each node implements the algorithm without too much information about the network) and low message complexity (i.e., communication overhead among nodes is low). This is the principle and objective of our algorithm design. In fact, regarding MCDS construction, Basagni *et al.* [15] have conducted extensive simulations which show that highly localized algorithms (protocols), such as Wu's algorithm [1] and Stojmenovic's algorithm [2], are rewarded with good performance. We expect that it would be the same case to MLCDS, i.e., highly localized algorithms would perform well. Inspired by Wu's and Stojmenovic's alogirthms, we propose the *L-MLCDS* algorithm to the MLCDS problem.

Similar to Wu's and Stojmenovic's algorithms[3], our *L-MLCDS* algorithm requires 2-hop neighborhood information and consists of two phases. Specifically, given a CRN, each CU (node) in the network should first collect its 2-hop neighborhood information. The information that each node exchanges includes: 1) its ID, 2) its available channels, and 3) its 1-hop neighbors' IDs and available channels. Such information collection could be realized by the nodes employing either common control channel (CCC) or rendezvous algorithms without using CCC. If a CCC is assumed to be available, the adjacent nodes can simply exchange information over the CCC. If no CCC is available, two adja-

[3] There is only one phase in a later improved version [24]. Each node v computes a subgraph induced by the 1-hop neighbors with higher keys than v itself. v is dominator if the subgraph contains a connected component which can cover all v's neigbhors, and is dominatee otherwise.

cent nodes can apply a rendezvous algorithm (e.g., the one in [20], [21] or [22]) to establish a communication link between them. Information exchange could be subsequently realized over this link. It should be emphasized that CCC or rendezvous algorithms are used only in building CDS. Once a CDS is constructed, the CDS can operate over available channels without CCC and rendezvous algorithms.

The first phase in our *L-MLCDS* algorithm is exactly the same as that in Wu's and Stojmenovic's algorithms. Specifically, a node determines itself to be a dominator if it has two neighbors which do not have a direct link between them; otherwise it will be a dominatee. All the dominators form a preliminary CDS of the network after the first phase [1]. This CDS is usually large in terms of size and its lifetime is small. Recall that three objectives are optimized sequentially in the MLCDS problem: the lifetime, the internal lifetime, and the size of CDS. In the second phase, we adopt the pruning mechanism to further reduce the size of CDS as well as improve the lifetime and internal lifetime of CDS. To maximize the lifetime of CDS, we should select the dominators which are associated with links of longer lifetime. To minimize the size of CDS, we should select the dominators which can dominate more neighbors.

To this aim, we carefully design a key for each node. Given node v, let $ID(v)$ be node v's ID and $\boldsymbol{N}(v)$ the set of node v's neighbors. Node v's key is defined as a two-tuple $key(v)=(f(v), ID(v))$, where $f(v)$ is defined as follows:

$$f(v)=med(v)\times|\boldsymbol{J}(v)|,\ med(v)=\text{median}\{\rho_{u,v}\,|\,u\in\boldsymbol{N}(v)\}$$
$$\text{and } \boldsymbol{J}(v)=\{u\,|\,\rho_{u,v}\geq med(v),\ u\in\boldsymbol{N}(v)\}.$$

$\rho_{u,v}$ is the lifetime of link $e_{u,v}$ (between nodes u and v), and $med(v)$ is the *median* value of the lifetimes of links with node v as an endpoint. $\boldsymbol{J}(v)$ denotes the set of v's neighbors such that the lifetime of link between v and each node in $\boldsymbol{J}(v)$ is not less than $med(v)$. Clearly, both $med(v)$ and $\boldsymbol{J}(v)$ take into account lifetimes of links. Since lifetime of a link reflects how the dynamic availabilities of channels on this link are considered (see Definition 1 in Section 3), the above defined key actually considers the dynamic availability of each channel which is a characteristic of CRNs. To some extent, $med(v)$ serves the objective of maximizing the lifetime of CDS, and $|\boldsymbol{J}(v)|$ serves the objective of minimizing the size of CDS. The proposed metric $f(v)$ aims to strike a balance between these two objectives. Intuitively, if a node has more links (i.e., neighbors) that are with longer lifetime, it will be assigned with a larger key and consequently has a greater chance to be selected as a dominator. The pruning process using the key is to minimize the size of the previously-constructed CDS as well as maximize the lifetime, particularly the internal lifetime, of the CDS.

However, the definition of key concerns the only links with lifetime greater than the median value, by which it may prune the node associated with links with smaller lifetime. If this node is the best choice

```
L-MLCDS Algorithm
/*Assume that each node has collected its 2-hop neighborhood information. Node v in the network executes the following: */
1: // The first phase is to construct a preliminary CDS
2: v is initialized as a "dominatee";
3: FOR each $u\in N(v)$
4:   FOR each $w\in\{w\,|\,w\in N(v), w\neq u\}$
5:     IF link $(u, w)$ does not exist
6:     // v has two unconnected neighbors
7:       v becomes a "dominator";
8:       BREAK; // exit the for-loops
9: // The second phase is to reduce the size of the CDS
10: IF v is a "dominator"
11:   Compute $key(v)$ and $key(u)$ for each $u$ in $N(v)$;
12:   Compute $h(\{v\}, N(v))$;
13:   Flag=FALSE;
14:   FOR each $u\in\{u\,|\,u\in N(v), key(u)>key(v)\}$
15:   // check Condition 1
16:     IF $\{u\}$ dominates $N[v]$
17:       Compute $h(\{u\}, N[v])$;
18:       IF $h(\{u\}, N[v])\geq h(\{v\}, N(v))$
19:         Flag=TRUE;
20:         BREAK; //exit the for-loop
21:   IF Flag= =FALSE
22:   // Condition 1 is not true, check Condition 2
23:     FOR each $u\in\{u\,|\,u\in N(v), key(u)>key(v)\}$
24:       FOR each $w\in\{w\,|\,w\in N(v), w\neq u$ and
                         $key(w)>key(v)\}$
25:         IF link $(u, w)$ exists and $\{u, w\}$ dominates $N[v]$
26:           Compute $h(\{u, w\}, N[v])$;
27:           IF $h(\{u, w\}, N[v])\geq h(\{v\}, N(v))$
28:             Flag=TRUE;
29:             BREAK; // exit the for-loops
30:   IF Flag= =TRUE    // Node v changes its role
31:     v becomes a "dominatee";
```

Fig. 7. Pseudo code of the *L-MLCDS* algorithm.

(with largest lifetime) to dominate some nodes in the network, pruning this node will definitely decrease the lifetime of CDS. Thus, we should strike a good balance by taking the links with small lifetime into account. To address this issue, we define the average lifetime of links between a node (say v) and the set of nodes dominated by v as

$$h(\{v\}, \boldsymbol{B})=\text{mean}\{\rho_{a,b}\,|\,a\in\{v\}, b\in\boldsymbol{B}\},$$

where *mean* denotes the operation of arithmetic average. Intuitively, $h(\{v\}, \boldsymbol{B})$ indicates the average strength (lifetime) that node v can dominate all nodes in $\boldsymbol{B}$. We can extend the definition to the general case of set $\boldsymbol{A}$ dominating set $\boldsymbol{B}$ as follows

$$h(\boldsymbol{A}, \boldsymbol{B})=\text{mean}\{\max\{\rho_{a,b}\,|\,a\in\boldsymbol{A}\}\ |\,b\in\boldsymbol{B}\}.$$

Notice that, given a node b in $\boldsymbol{B}$, multiple nodes in $\boldsymbol{A}$ may dominate b. We use $\max\{\rho_{a,b}\,|\,a\in\boldsymbol{A}\}$ to measure the strength (lifetime) that node b can be dominated by $\boldsymbol{A}$. $h(\boldsymbol{A}, \boldsymbol{B})$ indicates the average strength that node set $\boldsymbol{B}$ can be dominated by $\boldsymbol{A}$. Based on this definition, the second phase of pruning in our algorithm is as follows. A dominator, say v, will switch to be a dominatee if either of the following two conditions is met:

Condition 1. There is a neighbor of v, say u, such that 1) $\boldsymbol{N}[u] \supseteq \boldsymbol{N}[v]$ (i.e., $\{u\}$ can dominate $\boldsymbol{N}[v]$), 2) $key(u) > key(v)$, and 3) $h(\{u\}, \boldsymbol{N}[v]) \geq h(\{v\}, \boldsymbol{N}(v))$.

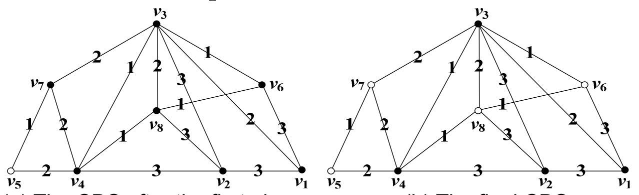
Condition 2. There are two neighbors of v, say u and w, such that 1) both are connected, 2) $\boldsymbol{N}[u] \cup \boldsymbol{N}[w] \supseteq \boldsymbol{N}[v]$ (i.e., $\{u, w\}$ can dominate $\boldsymbol{N}[v]$), 3) $\min\{key(u), key(w)\} > key(v)$, and 4) $h(\{u, w\}, \boldsymbol{N}[v]) \geq h(\{v\}, \boldsymbol{N}(v))$.

Notice that $\boldsymbol{N}[v] = \boldsymbol{N}(v) \cup \{v\}$, i.e., $\boldsymbol{N}[v]$ is the set of node v and its neighbors. Item (3) of Condition 1 could be interpreted as follows. If node u wants to "take over" node v (i.e., replace v to dominate v's neighbors and v), u should be at least as strong as v in terms of the lifetimes of the dominating links (on average). Similar explanation is to item (4) of Condition 2.

We formally present the pseudo code of the *L-MLCDS* algorithm in Fig. 7. In *L-MLCDS*, based on the 2-hop information, node v can determine the following values: 1) lifetimes of its links, 2) its key value $key(v)$, 3) $h(*,*)$ values in Conditions 1 and 2, and 4) lifetimes of its 1-hop neighbors' links and its 1-hop neighbors' key values.

5.2 Numerical Example

Before analyzing the proposed *L-MLCDS* algorithm, we demonstrate how it works by using the network in Fig. 6(a) as follows. Firstly, it can be identified that, when the first phase of the *L-MLCDS* algorithm ends, a preliminary CDS is formed by $\boldsymbol{S}=\{v_1, v_2, v_3, v_4, v_6, v_7, v_8\}$, as illustrated in Fig. 8(a). For example, node v_1 should determine to be a dominator since it has two neighbors, i.e., v_2 and v_6, which do not have a direct link between them. Notice that, the lifetime, the internal lifetime, and the size of $\boldsymbol{S}$ are 2, 2, and 7, respectively. Next, the size of CDS $\boldsymbol{S}$ is reduced and its lifetime is enhanced. Suppose that the ID of node v_i is i, i=1, 2, ⋯, 8. All nodes are sorted according to their keys in ascending order as: v_5, v_6, v_8, v_7, v_1, v_4, v_3, v_2. Take the sorting between nodes v_5 and v_6 as an example. Since $med(v_5)=\text{median}\{\rho_{5,7}, \rho_{5,4}\}=\text{median}\{1, 2\}=1.5$ and $\boldsymbol{J}(v_5)=\{v_4\}$, we have $f(v_5)=med(v_5)\times|\boldsymbol{J}(v_5)|=1.5\times1=1.5$ and $key(v_5)=(f(v_5), \text{ID}(v_5))=(1.5, 5)$. Similarly, we have $key(v_6)=(3, 6)$, Therefore, $key(v_5)$ is less than $key(v_6)$. During the second phase of pruning, dominator v_6 is removed from $\boldsymbol{S}$. Actually, node v_3 is a neighbor of v_6 such that $key(v_3)>key(v_6)$ and v_3 can dominate $\boldsymbol{N}[v_6]$. Furthermore, we have $h(\{v_3\}, \boldsymbol{N}[v_6])=\text{mean}\{\rho_{3,1}, \rho_{3,6}, \rho_{3,8}\}=\text{mean}\{2, 1, 2\}=5/3$ and $h(\{v_6\}, \boldsymbol{N}(v_6))=\text{mean}\{\rho_{6,1}, \rho_{6,3}, \rho_{6,8}\}=\text{mean}\{1, 3, 1\}=5/3$, that is, $h(\{v_3\}, \boldsymbol{N}[v_6]) \geq h(\{v_6\}, \boldsymbol{N}(v_6))$. Together with these facts, we see that Condition 1 is met for pruning dominator v_6. Similarly, we can identify that dominators v_8 and v_7 should also be pruned, with pruning condition (Condition 1) satisfied by nodes v_3 and v_4, respectively. No other dominators can be further pruned. For example, dominator v_1 will not be removed, since neither Condition 1 (v_3 can dominate $\boldsymbol{N}[v_1]$ but $h(\{v_3\}, \boldsymbol{N}[v_1])=2<h(\{v_1\}, \boldsymbol{N}(v_1))=8/3$) nor Condition 2 ($\{v_2, v_3\}$ can dominate $\boldsymbol{N}[v_1]$ but $h(\{v_2, v_3\}, \boldsymbol{N}[v_1])=5/2<h(\{v_1\}, \boldsymbol{N}(v_1))=8/3$) can be met in this case. Eventually, after pruning $\boldsymbol{S}$ becomes $\{v_1, v_2, v_3, v_4\}$, which is the final CDS generated by our *L-MLCDS* algorithm (see Fig. 8(b)). It can be identified that, the lifetime, the internal lifetime, and the size of CDS $\boldsymbol{S}$ are 2, 3, and 4, respectively. In contrast, Wu's algorithm generates the CDS $\{v_3, v_4, v_6, v_7, v_8\}$ of $\boldsymbol{G}$ and Stojmenovic's algorithm generates the CDS $\{v_3, v_4\}$ of $\boldsymbol{G}$, both having the same lifetime equal to 1.



(a) The CDS after the first phase. (b) The final CDS.

Fig. 8. A numerical example for the *L-MLCDS* algorithm.

5.3 Analysis of *L-MLCDS* Algorithm

In this section we analyze the correctness of our *L-MLCDS* algorithm which is based on Wu's algorithm and Stojmenovic's algorithm. However, regarding the correctness of the algorithms, no formal proof has been given for both Wu's algorithm [1] and Stojemenovic's algorithm [2]. Therefore, we first give formal proof of the correctness of these two algorithms. Notice that implementation of the three algorithms (i.e., Wu's, Stojmenovic's and ours) at each node is based on its 2-hop neighborhood information. In other words, each node is actually aware of only its local graph (2-hop neighborhood). Therefore, the input graph of the three algorithms should be the union of local graphs perceived by all nodes. We call such graph as a *perceptive graph* and denote it by $\boldsymbol{G}$. When $\boldsymbol{G}$ is connected but not completely connected (i.e., any two nodes are connected), the following lemma states that Wu's algorithm can generate a valid CDS of $\boldsymbol{G}$.

Lemma 6. *If perceptive graph* $\boldsymbol{G}$ *is connected but not completely connected (clique), Wu's algorithm generates a valid CDS of* $\boldsymbol{G}$*, regardless of the assumption of the physical layer (ideal physical layer or realistic physical layer).*

Proof. The proof is moved to the supplementary file. □

Notice that, any two nodes are connected when $\boldsymbol{G}$ is completely connected. Thus, no node will claim to be a dominator when the first phase of Wu's algorithm ends (A node becomes a dominator only when it has two neighbors which do not have a direct link between them). That is, the final set of dominators will be empty. This makes sense, as no CDS is needed to facilitate routing or broadcasting in this case [1]. Furthermore, in Wu's algorithm nodes are sorted according to their IDs. However, as long as each node is associated with a unique key (not limited to ID), Wu's algorithm can still work correctly. In other words, the correctness of Wu's algorithm does not rely on what key values are used to sort nodes.

Notice that, Stojmenovic's algorithm has the same first phase as Wu's algorithm. They differ only in the second phase of pruning. Stojmenovic's algorithm is normally more efficient than Wu's algorithm in terms of pruning dominators. However, when Wu's algorithm is allowed to use the same key values as Stojmenovic's algorithm and the idea physical layer is adopted, the pruning phases of both algorithms are equivalent (they prune the same set of dominators). This result is stated in the following lemma.

Lemma 7. *If the ideal physical layer is adopted, the pruning phases of Wu's and Stojmenovic's algorithms are equivalent, given that the same key values are used in both algorithms to sort nodes.*

Proof. The proof is moved to the supplementary file. □

According to Lemmas 6 and 7, we can verify the correctness of Stojmenovic's algorithm under the ideal physical layer. Notice that no message exchange is required in the pruning phase of Stojmenovic's algorithm. This implies that the pruning of Stojmenovic's algorithm would not be affected by the specific physical layer. Hence, if the realistic physical layer is adopted, Stojmenovic's algorithm can still output the valid CDS. Based on the correctness of Stojmenovic's algorithm, we can prove the correctness of *L-MLCDS*, stated in the following theorem.

Theorem 11. *If perceptive graph* ***G*** *is connected but not completely connected (clique), the L-MLCDS algorithm generates a valid CDS of* ***G****, regardless of the assumption of the physical layer (ideal physical layer or realistic physical layer).*

Proof. The proof is moved to the supplementary file. □

However, *L-MLCDS* is heuristic and does not guarantee global optimality or local optimality, in terms of the maximum lifetime and the minimum size of the CDS. Regarding message cost of *L-MLCDS*, similar to Wu's and Stojmenovic's algorithms, *L-MLCDS* requires each node to exchange messages with its 1-hop neighbors for collecting its 2-hop neighborhood information. Therefore, message cost of *L-MLCDS* is the same as that of Wu's and Stojmenovic's algorithms in the first phase. In the second phase, Wu's algorithm requires nodes to exchange information of their status (dominator or dominatee), while both *L-MLCDS* and Stojmenovic's algorithm do not need extra message exchange. In conclusion, in terms of message cost, performance of *L-MLCDS* is similar to that of Stojmenovic's algorithm but is better than that of Wu's algorithm.

6 SIMULATION

We build the simulator in Matlab R2011a to evaluate the performance of the two algorithms proposed by us to the MLCDS problem, namely, *C-MLCDS* (centralized algorithm) and *L-MLCDS* (localized algorithm). To the best of our knowledge, there is no existing work on computing CDS in CRNs. For comparison, we select four representative algorithms, Guha's [7], Wan's [9], Wu's [1], and Stojmenovic's [2], which were proposed for constructing MCDS (minimum CDS) in conventional wireless networks (e.g., ad hoc networks and wireless sensor networks). In Table 1, we highlight the characteristics of our algorithms and the four benchmark algorithms.

TABLE 1
CHARACTERISTICS OF DIFFERENT ALGORITHMS

Algorithm	Objective	Centralized/ Distributed	Information required
Guha's	Min size	Centralized	Global
Wan's	Min size	Distributed	Global
Wu's	Min size	Distributed	2-hop
Stojmenovic's	Min size	Distributed	2-hop
C-MLCDS	Max lifetime, Min size	Centralized	Global
L-MLCDS	Max lifetime, Min size	Distributed	2-hop

6.1 Simulation Setup

For comprehensive study, we define four types of *spectrum environments* (SEs) as suggested by Lee *et al.* in [5]. Each SE represents a possible combination of *PU's activity* and *spectrum opportunity*. Recall that each PU could behave in status of ON (active on the channel) or OFF (inactive on the channel). We assume that the sojourn times of the PU in ON status and OFF status are random variables which follow the exponential distributions [4] [5] with parameters μ and λ, respectively (Our proposed algorithms are *independent* of the distribution model of PU's activity). Accordingly, PU's activity can be classified into two cases: *low activity* and *high activity*. Specifically, it is called low activity if μ<1 and λ<1, and high activity if μ>1 and λ>1. Notice that $1/\mu$ and $1/\lambda$ are the expected sojourn times of PU being in ON status and OFF status, respectively. Therefore, low activity (μ<1 and λ<1) implies that the PU does not change its status frequently, while high activity (μ>1 and λ>1) means that the PU switches between ON status and OFF status frequently. Similarly, spectrum opportunity can be also classified into two cases: *low opportunity* and *high opportunity*. Specifically, it is called low opportunity if $P_{on}>P_{off}$, and high opportunity if $P_{on}<P_{off}$. Notice that $P_{on}=\mu/(\mu+\lambda)$ and $P_{off}=\lambda/(\mu+\lambda)$ are the probabilities that the PU is in ON status and OFF status in an instant of time, respectively [5]. Thus, low opportunity ($P_{on}>P_{off}$) implies that the PU is in ON status at most of the time, while high opportunity ($P_{on}<P_{off}$) means that the PU is in OFF status at most of the time. The four SEs are detailed in Table 2.

TABLE 2
FOUR TYPES OF SPECTRUM ENVIRONMENTS

Spectrum Environment	PU's Activity	Spectrum Opportunity	Parameters
I	Low	Low	$0<\mu<1$, $0<\lambda<1$, $\mu>\lambda$
II	Low	High	$0<\mu<1$, $0<\lambda<1$, $\mu<\lambda$
III	High	Low	$1<\mu<10$, $1<\lambda<10$, $\mu>\lambda$
IV	High	High	$1<\mu<10$, $1<\lambda<10$, $\mu<\lambda$

In our simulations, PUs and CUs are deployed in a

500m× 500m 2-dimension square. Locations of PUs and CUs are randomly generated according to the uniform distribution over the target area. The interference range of each PU is 300m, and the transmission range of each CU is 150m. We vary the number of CUs, i.e. *N*, from 20 to 140, and vary the number of PUs, i.e. *M*, from 10 to 50. As mentioned in Section 3, each PU operates over a unique channel and there are exactly *M* channels potentially available to the CUs.

For simplicity, the ideal physical layer is adopted in our simulations. We assume that CCC (common control channel) is employed in the network. Each CU can exchange messages with its neighbors over the CCC, and collect its 2-hop neighborhood information. The reason for adopting the ideal physical layer is two-fold. First, we have theoretically proved that, our *L-MLCDS* algorithm can generate valid CDS no matter which kind of physical layer (ideal or realistic) is adopted (see Theorem 11). Second, during the implementation of the *L-MLCDS* algorithm, each CU does not need to exchange messages with its neighbors (except initial message exchange for collecting 2-hop neighborhood information, which is also required in Wu's and Stojmenovic's algorithms). This means that the performance of our *L-MLCDS* algorithm is independent of the physical layer. To evaluate the performance of the algorithms, we focus on three metrics: 1) the lifetime of CDS, 2) the internal lifetime of CDS, and 3) the size of CDS.

In each simulation run, we randomly generate the locations of *M* PUs and *N* CUs according to the uniform distribution. Each PU randomly selects parameters μ and λ which are subject to the type of SE described in Table 2. With μ and λ, we can calculate $P_{\text{on}}=\mu/(\mu+\lambda)$ and $P_{\text{off}}=\lambda/(\mu+\lambda)$, and set the ON/OFF status of the PU according to the probabilities. Due to limited sensing capacity of CUs, the CUs usually cannot sense all the channels of the PUs [23]. We let each CU randomly select a portion of the total channels, say $p\times M$ channels (p is fixed at 60% in our simulations), as its candidate channels. Suppose that channel m is one of the candidate channels of CU i. Channel m is determined to be available to CU i if PU m (associated with channel m) is in OFF status or CU i is outside the interference range of PU m. After exchanging message mutually, two adjacent CUs can identify the set of their commonly-available channels. There is a communication link between these two CUs if the set is not empty. Though the lifetime of the link could be determined by Eq. (9) in the Appendix (see the supplementary file), we adopt the Monte Carlo method [16] in our simulations to estimate the lifetime of the link since it is time consuming to compute Eq. (9) when the number of CUs is large. For each link $e_{i,j}$, we randomly generate an instance of $x_{i,j}^{(m)}$ ($1\le m\le M$) which is the duration that channel m is being available on link $e_{i,j}$, according to the exponential distribution of PU m's activity. The longest duration of the channel being available on link $e_{i,j}$ is determined by $\max\{x_{i,j}^{(1)}, x_{i,j}^{(2)}, \cdots, x_{i,j}^{(M)}\}$. We generate 1000 instances to calculate a mean value to estimate the lifetime of link $e_{i,j}$, i.e., $\mathrm{E}(\max\{x_{i,j}^{(1)}, x_{i,j}^{(2)}, \cdots, x_{i,j}^{(M)}\})$. After each link and its lifetime are determined, we check whether the generated topology is connected. We drop disconnected topologies and keep only the connected topologies (valid instances). We run all the algorithms in these connected networks (L valid instances in total) to construct CDS. For each algorithm, we calculate the mean value of the lifetime, the internal lifetime, and the size of the CDS. In our simulations, we set L to be 200, i.e., the results reported in the following figures are the means of 200 separate runs.

6.2 Simulation Results

We evaluate the performance of the algorithms by using three metrics: 1) the lifetime of CDS, 2) the internal lifetime of CDS, and 3) the size of CDS. Accordingly, the simulation results consist of the following three parts. According to Theorem 5, the *C-MLCDS* algorithm gives the optimal lifetime and the optimal internal lifetime (with the optimal lifetime preserved). Instead of reporting the (internal) lifetime of the CDS, we show the percentage of the optimal (internal) lifetime achieved by each algorithm by using the results of *C-MLCDS* as a benchmark.

6.2.1 On Lifetime of CDS

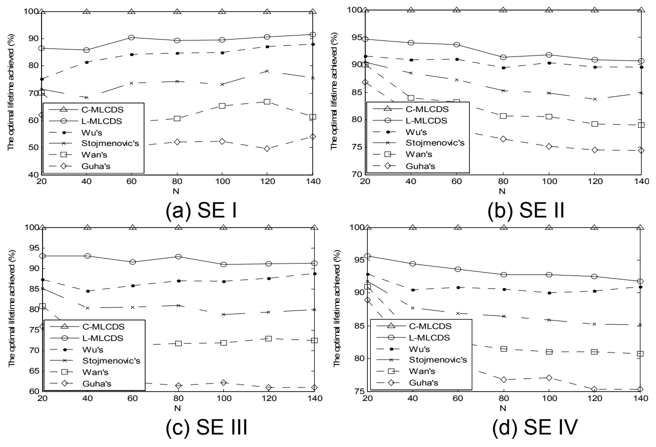


Fig. 9. Performance comparison of different algorithms in terms of the lifetime of CDS under four SEs, varying *N* (*M*=30).

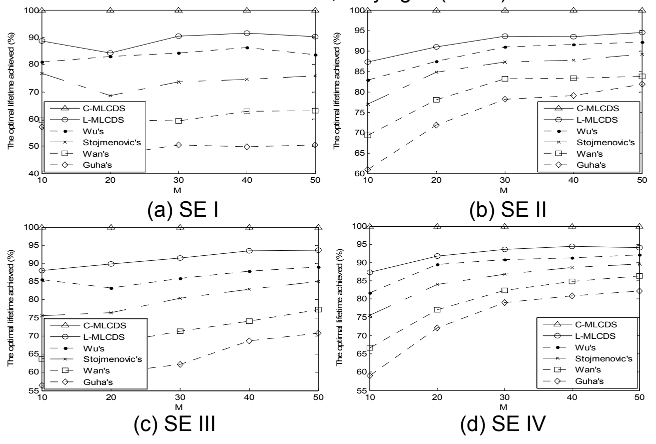


Fig. 10. Performance comparison of different algorithms in terms of the lifetime of CDS under four SEs, varying *M* (*N*=60).

Fig. 9 and Fig. 10 demonstrate the performance of different algorithms in terms of the lifetime of CDS under various SEs. In Fig. 9, we fix *M*, i.e., the number of channels, and report the lifetime of CDS against *N*, i.e., the number of CUs. In Fig. 10, contrastively, we fix *N* and report the lifetime of CDS against *M*. As we have pointed out, our *C-MLCDS* algorithm generates CDSs achieving 100 percent optimal lifetime. That is, *C-MLCDS* outperforms all the other algorithms. It is obvious that, among the five suboptimal algorithms, including Guha's, Wan's, Wu's, Stojmenovic's and our *L-MLCDS* algorithm, *L-MLCDS* has the best performance.

Specifically, when we focus on SE I and fix *M*, as shown in Fig. 9(a), *L-MLCDS* generates CDSs which steadily achieve 90% optimal lifetime or more when *N* is greater than 40. Following *L-MLCDS*, Wu's algorithm outperforms the other four algorithms and it can generate CDSs achieving around 82% optimal lifetime in most cases. Clearly, Stojmenovic's algorithm is inferior to both *L-MLCDS* and Wu's algorithm, generating CDSs achieving 70% to 75% optimal lifetime. Wan's algorithm generates CDSs with 60% to 65% optimal lifetime achieved, which are significantly lower than those by *L-MLCDS*. As for Guha's algorithm, it performs worst and generates CDSs with only 50% optimal lifetime achieved at most of the time. Similar performance comparison results of these algorithms can be obtained from other figures.

6.2.2 On Internal Lifetime of CDS

Fig. 11 and Fig. 12 demonstrate the performance of different algorithms in terms of the internal lifetime of CDS under various SEs. In Fig. 11, we fix *M* and report the internal lifetime of CDS against *N*; while in Fig. 12 we fix *N* and report the internal lifetime of CDS against *M*. Notice that, as the optimal baseline for comparison, our *C-MLCDS* algorithm generates CDSs with 100% optimal internal lifetime achieved. According to Figs. 11 and 12, we can see that the ranking of the other five algorithms is: *L-MLCDS*>Wu's>Stojmenovic's>Wan's>Guha's, in terms of the internal lifetime of CDS, which is the same to the ranking in terms of the lifetime of CDS. For example, when *M* is fixed, as shown in Fig. 11(d) (SE IV), *L-MLCDS* generates CDSs with at least 90% optimal internal lifetime achieved, significantly exceeding that of Wu's and Stojmenovic's algorithms (around 75%) and that of Wan's and Guha's algorithms (around 65%). Similar performance gap between *L-MLCDS* and these compared algorithms can be identified in other figures.

It can be observed that the curves in Figs. 11(a)-(b) do not have a clear trend as *N* increases. The reason is as follows. In simulation, we fix the network region as a 500m×500m 2-dimension square. Increasing *N* will increase node density which results in more links in the network. On the one hand, as the number of links increases (larger pool), the minimum lifetime of links decreases. Notice that the internal lifetime of a CDS is highly dependent on the minimum lifetime of links associated with the CDS nodes (dominators). The internal lifetime of the CDS would decrease as *N* increases. On the other hand, with more nodes and more links in a CDS, the CDS would become more robust and consequently has a longer internal lifetime. Therefore, the internal lifetime of the CDS shown in Figs. 11(a)-(b) is the resulting effect of these two contrary facts. This is the reason why the curves do not have a clear trend as *N* increases.

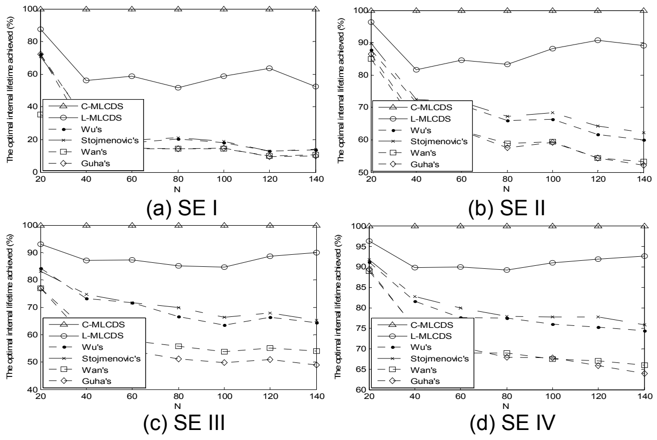


Fig. 11. Performance comparison of different algorithms in terms of the internal lifetime of CDS under four SEs, varying *N* (*M*=30).

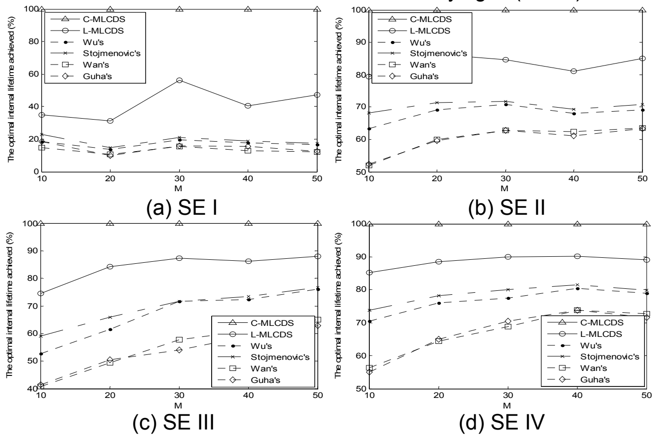


Fig. 12. Performance comparison of different algorithms in terms of the internal lifetime of CDS under four SEs, varying *M* (*N*=60).

Except for spectrum environments (SEs), all simulation parameters are same in Figs. 12 (a)-(d). We can see that the curves in Figs. 12(a)-(b) (using SE I and SE II) do not have a clear trend while those in Figs. 12(c)-(d) (using SE III and SE IV) have a clear trend. Therefore, selection of SE I and SE II is the cause of the unclear trend of curves in Figs. 12(a)-(b). According to Table 2, PU's activity is low in both SE I and SE II which implies that the PU does not change its status (ON/OFF) frequently. If a PU is currently in OFF status, it will probably stay in OFF status for a very long time which results in a long duration of its corresponding channel being available. In contrast, if a PU is currently in ON status, duration of its corresponding channel being available is short. Thus, there is a big variance in the durations of channels being available in SE I (Fig. 12(a)) and SE II (Fig. 12(b)). On the one hand, the big variance in the durations of channels would result in a big variance in lifetimes of different links. In this case, it is difficult to compute the CDS with the maximum internal lifetime due to huge searching space. So, the internal lifetime of the CDS computed by different algorithms would decrease in SE I (Fig. 12 (a)) and SE II (Fig. 12(b)). On the other hand, as *M* increases, there are more potentially available channels on each link. When number of available channels increases on each link, lifetimes of different links become closer. As a result, the internal lifetime of the CDS computed by different algorithms would increase since it is easier to find a CDS with a long/optimal lifetime in the network where links are associated with close/same lifetimes. The unclear trend of curves in Figs. 12(a)-(b) is the resulting effect of these two contrary facts.

6.2.3 On Size of CDS

Fig. 13 shows the performance of different algorithms in terms of the size of CDS against *N* (*M* is fixed as 30). We omit the results obtained by fixing *N* but varying *M*. Actually, the sizes of CDSs constructed by different algorithms hardly vary with *M*, and they are mostly affected by *N* (i.e., the number of CUs).

First of all, according to Fig. 13, the sizes of CDSs constructed by the algorithms increase as *N* increases. Among the four conventional CDS algorithms, Guha's algorithm (centralized) generates the CDSs with smallest size. Wan's algorithm (distributed with global information) outperforms both Wu's and Stojmenovic's algorithms (distributed with 2-hop neighborhood information). Furthermore, Stojmenovic's algorithm is superior to Wu's algorithm. Our observation on such performance difference of these three distributed algorithms is consistent with that identified by Basagni *et al.* for conventional wireless networks [15].

Our centralized *C-MLCDS* algorithm generates CDSs with size larger than Guha's algorithm and Wan's algorithm. Moreover, in many cases (see Figs. 13(c) and 13(d)), *C-MLCDS* even performs worse than Stojmenovic's algorithm. Since optimizing the size of CDS is the third objective with lower priority in our focused MLCDS problem, it is very likely for the proposed *C-MLCDS* algorithm to construct CDSs with relatively large size. Essentially, it may reflect the trade-off between the lifetime and the size of CDS when constructing proper virtual backbone in CRNs. As for our localized *L-MLCDS* algorithm, which is also based on 2-hop neighborhood information, it generates CDSs with size larger than Stojemenovic's algorithm. However, compared with Wu's algorithm, *L-MLCDS* usually performs better in terms of constructing slim CDSs. Actually, except for the scenario of SE I (see Fig. 13(a)), in other scenarios *L-MLCDS* generates CDSs with size smaller than Wu's algorithm (see Figs. 13(b), 13(c) and 13(d)). In this sense, our *L-MLCDS* algorithm is efficient in not only maximizing the lifetime of CDS but minimizing the size of CDS.

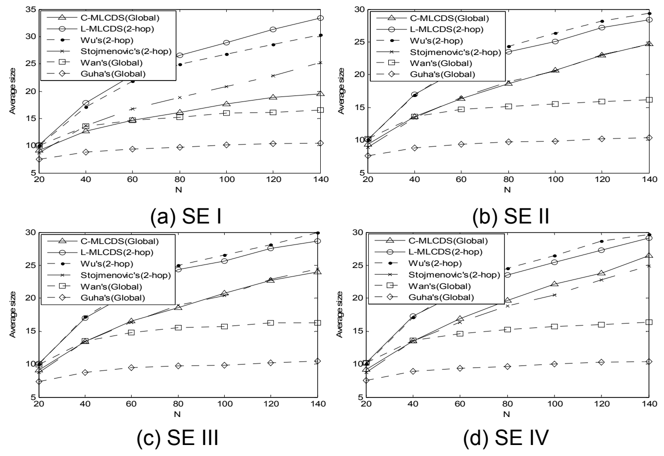


Fig. 13. Performance comparison of different algorithms in terms of the size of CDS under four SEs, varying *N* (*M*=30).

7 Future Work

In future work, other difinitions of lifetime, e.g., a probabilistic model of lifetime, could be explored. For example, each link is associated with a probability which indicates the robustness of one node dominating another. The probability could be meatured by considering activities of PUs on the channels of this link. Then, lifetime of a CDS is measured by the joint probability that this CDS is valid. In this definition, if a dominatee is covered by more dominators, it is more robust to dominate this dominatee while the size of the CDS would become larger.

Acknowledgment

This work is supported in part by General Research Fund of Hong Kong [No. GRF HKBU 210412] and Research Grant of Hong Kong Baptist University [No. FRG2/11-12/160]. Zhiyong Lin is supported by the National Natural Science Foundation of China [No. 61202453] and the Natural Science Foundation of Guangdong Province in China [No. S2011040002890]. The work of Ivan Stojmenovic was supported by NSERC CRDPJ 386874-09 (Reliable and secure QoS routing and transport protocols for mobile ad hoc networks), NSERC Discovery grant, Government of China for the Tsinghua 1000 Plan Distinguished Professor (2012-5) position, and NSFC Collaboration project number 61120106008.

References

[1] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," in *Proc. of the 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 7-14, 1999.
[2] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14-25, 2002.
[3] Y. Song and J. Xie, "A Distributed Broadcast Protocol in Multi-hop Cognitive Radio Ad Hoc Networks without a Common Control Channel," in *Proc. of IEEE INFOCOM 2012*, pp. 2273-2281, 2012.

[4] S. Geirhofer, L. Tong, and B.M. Sadler, "Dynamic Spectrum Access in the Time Domain: Modeling and Exploiting White Space," *IEEE Communications Magazine*, vol. 45, no. 5, pp. 66-72, 2007.
[5] W.-Y. Lee and I.F. Akyildiz, "Optimal Spectrum Sensing Framework for Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 10, pp. 3845-3857, 2008.
[6] Z. Lin, H. Liu, X. Chu, Y.-W. Leung, and I. Stojmenovic, "Maximizing Lifetime of Connected-Dominating-Set in Cognitive Radio Networks," in *Proc. of IFIP Networking 2012*, Part II, pp. 316-330, 2012.
[7] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, vol. 20, pp. 374-387, 1998.
[8] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K.-I. Ko, "A Greedy Approximation for Minimum Connected Dominating Sets," *Theoretical Computer Science*, vol. 329, pp. 325-330, 2004.
[9] P.-J. Wan, K.M. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Sets in Wireless Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 9, no. 2, pp. 141-149, 2004.
[10] Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, and D.-Z.Du, "On Greedy Construction of Connected Dominating Sets in Wireless Networks," *Wireless Communications and Mobile Computing*, vol. 5, pp. 927-932, 2005.
[11] R. Misra and C. Mandal, "Minimum Connected Dominating Set Using a Collaborative Cover Heuristic for Ad Hoc Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 292-302, 2010.
[12] K. Sakai, C.-H. Huang, W.-S. Ku, M.-T. Sun, and X. Cheng, "Timer-Based CDS Construction in Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 10, pp. 1388-1402, 2011.
[13] Y. Wu and Y. Li, "Connected Dominating Sets," in *Handbook of Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*, H. Liu, Y. W. Leung, and X. Chu, Eds., Bentham Science, pp.19-39, 2009.
[14] I. Stojmenovic, A. Nayak, J. Kuruvila, F. Ovalle-Martinez, and E. Villanueva-Pena: "Physical Layer Impact on the Design and Performance of Routing and Broadcasting Protocols in Ad Hoc and Sensor Networks," *Computer Communications*, vol. 28, no. 10, pp. 1138-1151, 2005.
[15] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, "Localized Protocols for Ad Hoc Clustering and Backbone Formation: a Performance Comparison," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 4, pp. 292-306, 2006.
[16] G. S., Fishman: Monte Carlo: Concepts, Algorithms, and Applications, New York: Springer, 1995.
[17] T. Jiang, T. Li, and J. Ren: "Towards Secure Cognitive Communications in Wireless Networks," *IEEE Wireless Communications*, to appear.
[18] P. Slavík, "A Tight Analysis of the Greedy Algorithm for Set Cover," *Journal of Algorithms*, vol. 25, pp. 237-254, 1997.
[19] I. Akyildiz, W.-Y. Lee, and K. R. Chowdhury, "CRAHNs: Cognitive Radio Ad Hoc Networks," *Ad Hoc Networks*, vol. 7, no. 5, pp. 810-836, 2009.
[20] Z. Lin, H. Liu, X. Chu, and Y.W. Leung, "Jump-Stay Based Channel-Hopping Algorithm with Guaranteed Rendezvous for Cognitive Radio Networks," in *Proc. of IEEE INFOCOM 2011*, pp. 2444-2452, 2011.
[21] H. Liu, Z. Lin, X. Chu, and Y.-W. Leung, "Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1867-1881, 2012.
[22] H. Liu, Z. Lin, X. Chu, and Y.-W. Leung, "Ring-Walk Based Channel-Hopping Algorithms with Guaranteed Rendezvous for Cognitive Radio Networks," in *Proc. of WiSARN2010-FALL*, pp.755-760, 2010.
[23] T. Yücek and H. Arslan, "A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications," *IEEE Communications Surveys &Tutorials*, vol. 11, no. 1, pp. 116-130, 2009.
[24] I. Stojmenovic and J. Wu, "Broadcasting and Activity Scheduling in Ad Hoc Networks," in *Mobile Ad Hoc Networking*, S. Basagni, M. Conti, S. Giordana and I. Stojmenovic, Eds., IEEE/Wiley, pp. 205-229,2004.

Zhiyong Lin received the BSc and MSc degrees both in applied mathematics from South China University of Technology, in 1999 and 2002, respectively. He received the PhD degree in computer application technology from South China University of Technology in 2009. Currently, he is an associate professor in the Dept of Computer Science, GuangDong Polytechnic Normal University. His research interests include algorithm design and analysis, machine learning, and computational intelligence.

Hai Liu received the BSc and MSc degrees in applied mathematics from South China University of Technology, in 1999 and 2002, respectively. He received the PhD degree in computer science from City University of Hong Kong in 2006. He is currently a Research Assistant Professor with the Department of Computer Science, Hong Kong Baptist University. His research interests include wireless networking, mobile computing, and algorithm design and analysis. He is a member of the IEEE.

Xiaowen Chu received the BEng degree in computer science from Tsinghua University, P.R. China, in 1999, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. Currently, he is an associate professor in the Department of Computer Science, Hong Kong Baptist University. His research interests include distributed and parallel computing and wireless networks. He is a senior member of the IEEE.

Yiu-Wing Leung received his B.Sc. and Ph.D. degrees from the Chinese University of Hong Kong in 1989 and 1992 respectively. He has been working in the Department of Computer Science of the Hong Kong Baptist University and now he is a full professor. His research interests include two major areas: 1) networking and multimedia which include the design and optimization of wireless networks, optical networks and multimedia systems, and 2) cybernetics and systems engineering which include evolutionary computing and multiobjective programming. He has published more than 70 journal papers in these areas, and most of which were published in various IEEE journals.

Ivan Stojmenovic received his Ph.D. degree in mathematics. He is Full Professor at the University of Ottawa, Canada. He held regular and visiting positions in Serbia, Japan, USA, Canada, France, Mexico, Spain, UK (as Chair in Applied Computing at the University of Birmingham), Hong Kong, Brazil, Taiwan, China and Australia. He published over 300 different papers, and edited seven books on wireless, ad hoc, sensor and actuator networks and applied algorithms with Wiley. He is editor-in-chief of IEEE Transactions on Parallel and Distributed Systems (2010-3), and founder and editor-in-chief of three journals. He is editor of over dozen journals (including IEEE Network, IEEE Transactions on Cloud Computing and ACM Wireless Networks) and steering committee member of IEEE Transactions on Emergent Topics in Computing. Stojmenovic is one of about 250 computer science researchers with h-index >50, has top h-index in Canada for mathematics and statistics, and has >13000 citations. He received four best paper awards and the Fast Breaking Paper for October 2003, by Thomson ISI ESI. He received the Royal Society Research Merit Award, UK (2006), and Humboldt Research Award, Germany (2012). He is Tsinghua 1000 Plan Distinguished Professor (2012-5). He is Fellow of the IEEE (Communications Society, class 2008), and Canadian Academy of Engineering (since 2012), and Member of the Academia Europaea (The Academy of Europe), from 2012 (section: Informatics). He was IEEE CS Distinguished Visitor 2010-11 and received 2012 Distinguished Service award from IEEE ComSoc Communications Software TC. He received Excellence in Research Award of the University of Ottawa 2009. Stojmenovic chaired and/or organized >60 workshops and conferences, and served in >200 program committees. He was program co-chair at IEEE PIMRC 2008, IEEE AINA-07, IEEE MASS-04&07, founded several workshop series, and is/was Workshop Chair at IEEE ICDCS 2013, IEEE INFOCOM 2011, IEEE MASS-09, ACM Mobihoc-07&08.